Δημήτρης Πρίτσος

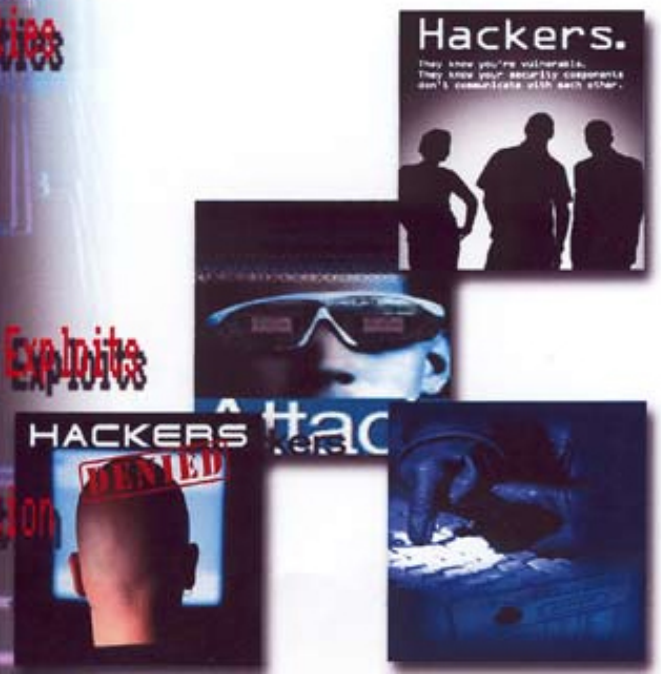
Εντοπισμός επιθέσεων κακόβουλων χρηστών που βασίζονται σε αδυναμίες υπερχείλισης μνήμης (Buffer Overflow)

Buffer Overflow Vulnerabilities

Buffer Overflow Exploits

Detection Of Buffer Oveflow Exploits

B.O.F. Detection Implementation



Αυτή η Πτυχιακή Εργασία πραγματοποιήθηκε σε συνεργασία του **Τεχνολογικού Εκπαιδευτικού Ιδρύματος Αθήνας** και του **Internet Systematics Lab** του **Εθνικού Ερευνητικού Κέντρου «Δημόκριτος».**
Αποτελεί μέρος της ολοκλήρωσης των σπουδών του συγγραφέα **Δημητρίου Πρίτσου** στο **Τμήμα Πληροφορικής του ΤΕΙ Αθήνας** και κομμάτι ευρύτερου ερευνητικού προγράμματος στον τομέα της *Ασφάλειας*, στο οποίο ο ίδιος λαμβάνει μέρος και πραγματοποιείται στο **Internet Systematics Lab του ΕΚΕΦΕ Δημόκριτος.**

Συγγραφέας
Δημήτριος Πρίτσος
Τελειόφοιτος Φοιτητής του Τμήματος Πληροφορικής του ΤΕΙ Αθήνας και μέλος του Internet Systematics Lab του ΕΚΕΦΕ «Δημόκριτος».

Εισηγητής Καθηγητής του ΤΕΙ Αθήνας

Δρ. Ιωάννης Χάλαρης

Υπεύθυνος εργαστηρίου Internet Systematics Lab

Δρ. Ιωάννης Κοροβέσης

ΠΡΟΛΟΓΟΣ

Αναμφισβήτητα σήμερα(2003) το πρόβλημα των επιθέσεων που βασίζονται στην Αδυναμία Υπερχείλισης της Μνήμης ή Buffer Overflow Vulnerability των εφαρμογών είναι το μεγαλύτερο πρόβλημα ασφάλειας των υπολογιστών της τελευταίας δεκαπενταετίας. Συγκεκριμένα σήμερα το 35% (http://www.lodoga.co.uk/attackinfo/thethreat/statistics.htm) περίπου των επιθέσεων σε ένα υπολογιστικό σύστημα που είναι συνδεμένο στο Internet βασίζεται σε αυτή την αδυναμία. Το αξιοσημείωτο είναι ότι αυτό το πρόβλημα είχε παρουσιαστεί από το 1960 (βλέπε αναφορά στο άρθρο http://news.com.com/2100-1001-233483.html?legacy=cnet&tag=st.ne.1002.bgif%3fst.ne.fd.gif.l) όταν η πολυτέλεια της χρήσης του υπολογιστή ήταν για τους λίγους, ενώ η κύρια χρήση του υπολογιστή ήταν για την επίλυση σύνθετων μαθηματικών προβλημάτων. Αν και το πρόβλημα ήταν ιδιαίτερα γνωστό στους προγραμματιστές τότε έγινε η επιλογή της συμβίωσης με αυτό.

Η επιλογή της συμβίωσης ήταν επιλογή ανάγκης γιατί **το κόστος σε υπολογιστική ισχύ** για ελέγχους που θα εμπόδιζαν την εκδήλωση αυτής της αδυναμία στο λογισμικό ήταν τεράστιο για τα δεδομένα της εποχής. Η ανάγκη για **επιδόσεις(Performance)** ανάγκασε όλες τις γενιές των προγραμματιστών και αναλυτών να παραβλέπουν το θέμα της ασφάλειας όταν είχαν να

αντιμετωπίσουν το κρίσιμο θέμα των **επιδόσεων**. Δυστυχώς αυτή η επιλογή ήταν μοιραία για ένα σύνθετο σύστημα όπως είναι τα σημερινά πληροφοριακά συστήματα.

Πόσο μοιραία ήταν αυτή η επιλογή αποδείχτηκε με την εμφάνιση του πρώτου **σκουληκιού(Worm) του Internet** το 1987 που σχεδιάστηκε από τον **Morris** και η εξάπλωση του βασιζόταν σε **αδυναμίες Buffer Overflow** των τότε UNIX συστημάτων. Το Worm αυτό έδειξε ότι η **αδυναμία Buffer Overflow** των υπολογιστικών συστημάτων μπορεί να γίνει πολλές φορές πιο καταστροφική για τα ίδια τα συστήματα όταν αυτά **δικτυώνονται**. Με την εμφάνιση του Worm του Morris αποδείχτηκε ότι οι αδυναμίες Buffer Overflow θα ήταν ένα μεγάλο πρόβλημα **ασφάλειας** για τα πληροφοριακά συστήματα του internet μέχρι αυτές να απαλειφθούν εντελώς.

Έτσι ένα από τα πάγια θέματα που απασχολούν τον τομέα της **Ασφάλειας των Υπολογιστών** είναι να αντιμετωπίσει τέτοια προβλήματα όπως την εκμετάλλευση των αδυναμιών Buffer Overflow δηλαδή τα **Buffer Overflow Exploits**. Οι προσπάθειες της λύσης του προβλήματος είναι πολλές και το αντιμετωπίζουν από διάφορες πλευρές.

Παρά τις διάφορες προσπάθειες το πρόβλημα των αδυναμιών Buffer Overflow είναι ένα θέμα που δεν έχει λυθεί ακόμα. Οι λύσεις που έχουν προταθεί κατά καιρούς σε αυτό το πρόβλημα είναι πολλές και **συνήθως προσπαθούν να αντιμετωπίσουν μόνο ορισμένες πλευρές του προβλήματος** και όχι να το εξαλείψουν ολοκληρωτικά. Αυτό φυσικά συμβαίνει γιατί η μοναδική σίγουρη λύση που γνωρίζουμε μέχρι σήμερα είναι να ξαναγραφτούν οι βασικές εφαρμογές από την αρχή κάτι που προς το παρών αποφεύγεται για διάφορους λόγους. Ο βασικότερος λόγος είναι η πολυπλοκότητα της λύσης αυτής και η άμεση συνέπεια του που είναι το **οικονομικό κόστος**.

Γενικά τις διάφορες λύσεις που έχουν προταθεί κατά καιρούς για το πρόβλημα μπορούμε να τις χωρίσουμε σε δύο μεγάλες κατηγορίες. Στην πρώτη κατηγόρια ανήκουν οι μέθοδοι που προσπαθούν να κλείσουν τις **αδυναμίες Buffer Overflow** ώστε να μην μπορεί να συμβεί κάποιο Buffer Overflow Exploit. Στην δεύτερη κατηγορία ανήκουν οι μέθοδοι που προσπαθούν να εμποδίσουν τα ίδια τα **Buffer Overflow Exploit** να βλάψουν το σύστημα.

Οι λύσεις της πρώτης κατηγορίας έχουν σαν κύριο στόχο να απαλείφουν τα **Buffer Overflow Vulnerabilities** κατά την παραγωγή μίας εφαρμογής. Μερικές από τις πιο συνηθισμένες λύσης αυτής της κατηγορίας αναφέρονται στο **κεφάλαιο 3** αυτής της πτυχιακής. Πολύ λίγες από αυτές τις λύσεις έχουν χρησιμοποιηθεί μέχρι σήμερα στην παραγωγή και ο βασικός λόγος είναι ότι μπορούν να προσφέρουν λογισμικό χωρίς αδυναμίες **Buffer Overflow** μόνο αν μία εφαρμογή ξανασχεδιαστεί από την αρχή και μόνο αν σε αυτή **δεν χρησιμοποιηθεί παλιός κώδικας**.

Η δεύτερη μεγάλη κατηγορία αντιμετώπισης **των Buffer Overflow Exploit** είναι αυτή που έχει ιδιαίτερο ενδιαφέρον. Στην κατηγορία αυτή ανήκει κάθε μέθοδος που σαν στόχο έχει τον **εντοπισμό(Detection) των ίδιων των Buffer Overflow Exploits** και όχι των αδυναμιών **ευάλωτου(Vulnerable) λογισμικού** σε τέτοιου είδους επιθέσεις. Το βασικό χαρακτηριστικό που κάνει όλες αυτές τις λύσεις να έχουν ιδιαίτερο ενδιαφέρον είναι ότι αυτές προσπαθούν

να εμποδίσουν τα **Buffer Overflow Exploits** αφού ένα σύστημα έχει μπει σε παραγωγική διαδικασία. Η πιο γνωστή λύση αυτή της κατηγορίας είναι τα Anti-Virus που εντοπίζουν εκτός από διάφορες άλλες οικογένειες επιθέσεων και αυτή των Buffer Overflow Exploits. Το βασικό μειονέκτημα που έχουν αυτές οι λύσεις μέχρι σήμερα είναι ότι δεν μπορούν να εμποδίσουν τα **Buffer Overflow Exploits** πριν αυτά γνωστοποιηθούν.

Ο σκοπός αυτής της πτυχιακής είναι να παρουσιάσει μία νέα μέθοδο αντιμετώπισης των **Buffer Overflow Exploits** και την εφαρμογή της που βασίζεται σε μία τεχνική που ονομάζεται Abstract Execution of Payload(A.E.P.). Η μέθοδος αυτή έχει σαν στόχο να εντοπίζει τα **Buffer Overflow Exploit** και μάλιστα **πριν αυτά γνωστοποιηθούν**. Η εφαρμογή της μεθόδου αυτής που έγινε από τον συγγραφέα της πτυχιακής, είναι ένα **module** του **Network Intrusion Detection System (NIDS)** που λέγεται **Snort**. Για να μπορέσει να γίνει κατανοητή η εφαρμογή αυτής της άλλα και η ίδια η μέθοδος χρειάζεται να γίνει προηγουμένως κατανοητό τι είναι ένα Buffer Overflow Exploit καθώς και τα ιδιαίτερα χαρακτηριστικά του.

Για να εξυπηρετηθεί αυτός ο σκοπός, η εργασία αυτή χωρίζεται σε δύο μέρη. Το πρώτο μέρος έχει σαν βασικό στόχο να δώσει στον αναγνώστη όσο το δυνατόν ποιο σφαιρική εικόνα για τα Buffer Overflow Exploits καθώς και τα βασικότερα από τα χαρακτηριστικά που δυσκολεύουν τον εντοπισμό τους από συνηθισμένα εργαλεία όπως τα Anti-Virus. Το δεύτερο μέρος παρουσιάζει αναλυτικά την μέθοδο του Abstract Execution of Payload ώστε να εντοπίζονται τα Buffer Overflow Exploits αλλά και πώς καταλήγουμε σε αυτή την μέθοδο. Επίσης στο δεύτερο μέρος παρουσιάζεται η υλοποίηση της μεθόδου εντοπισμού A.E.P σαν Module του Snort 2.0.

Συγκεκριμένα στο **Κεφάλαιο 1** γίνεται μία επισκόπηση (Overview) των αιτιών που οφείλονται τα **Buffer Overflow Vulnerabilities** δηλαδή οι αδυναμίες υπερχειλίσεις αλλά και πώς χτίζονται τα **Buffer Overflow Exploits**, από τους κακόβουλους χρήστες (Hackers), με σκοπό να εκμεταλλευτούν αυτές τις αδυναμίες.

Στο **Κεφάλαιο 2** γίνεται μια αναλυτικότερη περιγραφή για το πώς χτίζονται τα Buffer Overflow Exploit με σκοπό να παρουσιαστούν όλες οι σημαντικότερες ιδιαιτερότητες τους ώστε να γίνει κατανοητή η μέθοδος εντοπισμού **A.E.P**(Abstract Execution of Payload). Συγκεκριμένα παρουσιάζεται μέσα από την διαδικασία χτισίματος των Buffer Overflow Exploit γιατί το καθένα από αυτά θα μπορούσε να χαρακτηριστεί σαν «μοναδικό». Η μοναδικότητα αυτή οφείλεται στην ανάγκη να ληφθούν υπόψη ένα μεγάλο σύνολο από παραμέτρους όπως οι ιδιαιτερότητες της εφαρμογής που θα δεχτεί την επίθεση ή την πλατφόρμα που αυτή η εφαρμογή εκτελείται.

Η βασικές έννοιες που παρουσιάζονται στο κεφάλαιο 2 είναι οι **ακολουθίες RET, το ShellCode** και **το Sledge**. Στην αρχή περιγράφονται **οι ακολουθίες RET ή RA (Return Address)** και το **ShellCode** που αποτελούν την καρδιά ενός B.O.E. Στην συνέχεια γίνεται μια αναλυτική περιγραφή **του τμήματος ενός Buffer Overflow Exploit που λέγεται Sledge** και η χρησιμότητα του. Επίσης μέσο απλών παραδειγμάτων τεκμηριώνεται πόσο σημαντικό είναι το Sledge για να αυξηθεί η πιθανότητα επιτυχίας του B.O.E αλλά και πότε αυτό είναι

περιττό. Εκτός από τις τρείς αυτές βασικές έννοιες παρουσιάζονται οι μέθοδοι που οι Hacker **διαμορφώνουν τα BOE ώστε να αποφεύγουν διάφορα εμπόδια όπως τα φίλτρα των εφαρμογών ή Anti-Virus**. Τέλος στο **Κεφάλαιο 2** παρουσιάζονται τα Heap Based Buffer Overflow που είναι η πιο περίπλοκη μορφή των Buffer Overflow Exploits.

Στο **Κεφάλαιο 3** γίνεται μια σύντομη επισκόπηση των πιό συνηθισμένων μεθόδων αντιμετώπισης των Buffer Overflow Exploit αλλά και των Buffer Overflow Vulnerabilities. Οι μέθοδοι αυτοί δεν παρουσιάζονται με σκοπό να συγκριθούν με την τεχνική εντοπισμού A.E.P. Ο σκοπός αυτού του κεφαλαίου είναι να δώσει μια γενική εικόνα για τις προσπάθειες που έχουν γίνει για την αντιμετώπιση του προβλήματος και να γίνει κατανοητό ότι καμία από τις προσπάθειες αυτές δεν λύνει το πρόβλημα συνολικά.

Στο **Κεφάλαιο 4** γίνεται μια αναλυτική περιγραφή αφενός του ίδιου τον αλγόριθμο **Abstract Execution of Payload** και το πως ο αλγόριθμος αυτός χρησιμοποιείται σαν **μέθοδος εντοπισμού των Buffer Overflow Exploits**, αφετέρου παρουσιάζεται πώς καταλήγουμε σε αυτή την μέθοδο. Συγκεκριμένα παρουσιάζεται η έννοια της **«εκτελεσιμότητας»** και πώς αυτή είναι το βασικό κριτήριο που χρησιμοποιεί ο αλγόριθμος A.E.P για να εντοπίζει τα B.O.E. Η **«εκτελεσιμότητα»** είναι ένα χαρακτηριστικό που παρουσιάζεται έντονα στα πακέτα των δικτύων που περιέχουν **B.O.E** και συγκεκριμένα οφείλεται στο **sledge** τους.

Στην ανάλυση του Αλγορίθμου A.E.P παρουσιάζεται η εξάρτιση του από δύο βασικούς παράγοντες το **Instruction Set του επεξεργαστή** του υπολογιστικού συστήματος, αλλά και το **Threshold**. Η εξάρτηση του από το πρώτο είναι μοιραία γιατί μόνο έτσι μπορεί να εξεταστεί αν παρουσιάζεται «εκτελεσιμοτητα» και σε τι βαθμό μέσα στο Payload ενός πακέτου. Στην συνέχεια παρουσιάζεται η ανάγκη του **Threshold** δηλαδή η ανάγκη για ένα **Κατώφλι** που θα διαχωρίζει τα πακέτα που περιέχουν Buffer Overflow Exploits από αυτά που δεν περιέχουν. Όπως θα δει αναλυτικά ο αναγνώστης η ανάγκη για Threshold προκύπτει από το γεγονός ότι ακόμα και σε πακέτα που δεν περιέχουν εκτελέσιμες εντολές του επεξεργαστή (Instruction Set) παρουσιάζεται έστω και σε μικρό βαθμό η εκτελεσιμότητα. Επίσης παρουσιάζεται η **σύμπτωση** της ύπαρξης της «εκτελεσιμότητας» σε πακέτα που δεν έχουν BOE όπου από αυτή προκύπτει η ανάγκη του Threshold.

Επίσης στο **Κεφάλαιο 4** παρουσιάζονται μερικές πειραματικές μετρήσεις που έχουν γίνει στον αλγόριθμο A.E.P και πώς από αυτές τις μετρήσεις αποδεικνύεται ότι **με το κατάλληλο Threshold είναι δυνατόν** να χρησιμοποιηθεί αυτή η μέθοδος για να εντοπίζει τα Buffer Overflow Exploits με πολύ **καλά αποτελέσματα**. Ακόμα παρουσιάζονται οι δυσκολίες της ενσωμάτωσης του αλγόριθμου A.E.P σαν Module του Snort 2.0 ή οποιουδήποτε άλλου NIDS. Τέλος σε αυτό το κεφάλαιο παρουσιάζεται το πώς αυτή η μέθοδος είναι πιθανόν να έχει αρκετά μεγάλη επιτυχία ακόμα και σε ειδικές περιπτώσεις B.O.E.

Τέλος στο **κεφάλαιο 5** παρουσιάζεται αναλυτικά η δομή του Snort Module που κάνει τον εντοπισμό των Buffer Overflow Exploits μέσα σε ένα δίκτυο. Για να καταλάβει κάποιος πώς δουλεύει αυτό το Module στο κεφάλαιο αυτό γίνεται μια σύντομη παρουσίαση του Snort 2.0. Το Snort είναι ένα **Open Source NIDS** δηλαδή ένα **Network Intrusion Detection System**. Το Snort όπως και όλα τα NIDS έχει σαν βασικό ρόλο να παρακολουθεί τα πακέτα ενός

δικτύου με σκοπό να ειδοποιεί τον Administrator ενός δικτύου όταν το δίκτυο του δέχεται κάποια επίθεση. Τέλος στο κεφάλαιο αυτό παρουσιάζεται ο πηγαίος κώδικας του Module με αναλυτικά σχολεία.

Buffer Overflow & Buffer Overflow Exploit Overview

1.1 Γενικά

1.2 Το Buffer Overflow

1.2.1 Γενικά

1.2.2 Το Stack Buffer Overflow

1.3 Η γενική λειτουργία των υπολογιστικών συστημάτων

1.3.1 Γενικά

1.3.2 Η Εκτέλεση του προγράμματος και οι καταχωριτές

1.3.3 Η στοίβα και η χρήση της

1.3.4 Σύνοψη για την γενική λειτουργία του υπολογιστή

1.4 Πώς γίνονται τα Buffer Overflow Exploits

1.4.1 Γενικά

1.4.2 Αλλαγή της φυσιολογικής ροή του προγράμματος αντικαθιστώντας την Return Address.

1.4.3 Η μορφή του κώδικα που θα ενσωματωθεί στο Buffer Overflow String.

1.4.4 Χτίζοντας το πρώτο Buffer Overflow Exploit.

1.5 Πώς γίνονται στην πράξη τα Buffer Overflow Exploits

1.6 Αυξάνοντας την πιθανότητα επιτυχίας ενός B.O.E με την χρήση του NOP

1.7 Συνοπτικά

Buffer Overflow Exploits

2.1 Γενικά

2.2 Η μοναδικότητα ενός B.O.E

2.2 Τα βήματα για να χτιστεί ένα Buffer Overflow

2.3 Ανακαλύπτοντας το Buffer προς εκμετάλλευση.

2.3.1 Η αδυναμία των DLL

2.3.2 Η αδυναμία της stdio.h

2.3.3 Συμπέρασμα

2.4 Ο Εντοπισμός της RET.

2.4.1 Πια πρέπει να είναι η RET

2.4.2 Που πρέπει να μπει η RET μέσα στο B.O.E String

2.5 Ο Σχεδιασμός του Shell Code

2.6 Το περιεχόμενο του Sledge

2.6.1 Γενικά

2.6.2 Πότε υπάρχει ανάγκη για το Sledge

2.6.3 Όταν το Sledge χρειαστεί

2.7 Ειδικές συνθήκες που πρέπει να αντιμετωπίσει το BOE

2.7.1 Όταν το Buffer είναι μικρό

2.7.2 Παράκαμψη Φίλτρων της εφαρμογής

2.8 Buffer Overflow Exploits στον σωρό(Heap) και όχι στην στοίβα(Stack)

2.8.1 Γενικά

2.8.1 Η γενική ιδέα του Heap Based B.O.E

2.8.1.1 Η βασική δυσκολία

2.8.1.2 Η βασικές γνώσείς για τα Heap Based BOE

2.8.1.3 Περιπτώσεις επιτυχίας Heap BOE

2.8.2 Heap Based B.O.E που αλλάζουν ένα Pointer προς συνάρτηση με σκοπό να εκτελεστεί αυθαίρετος(arbitrary) κώδικας.

2.8.2.1 Η χρησιμότητα του Pointer προς συνάρτηση

2.8.2.2 Παράδειγμα BOE που στρέφει έναν Pointer προς συνάρτηση προς τον ShellCode

2.9 Διαφορά Local και Remote exploit

2.9.1 Local(τοπικά) BOE

2.9.2 Remote(απομακρυσμένα) BOE

2.10 Σύνοψη - Παρατηρήσεις

Μέθοδοι αντιμετώπισης των Buffer Overflow Exploits

3.1 Γενικά

3.2 Μέθοδοι πρόληψης του Buffer Overflow

3.2.1 Συγγραφή secure code

3.2.2 Dynamic run-time checks Libsafe

3.2.3 Automated Detection of Buffer Overflow via Static Analysis

3.2.4 Brute-force Analysis

3.3 Μέθοδοι αντιμετώπισης του Buffer Overflow μέσω της προστασίας κρίσιμων πληροφοριών

3.3.1 StackGuard

3.3.2 Visual C/C++ compiler του .NET Studio της Microsoft

Ειδική περίπτωση(Case): OpenSSL remote exploit

ΒΙΒΛΙΟΓΡΑΦΙΑ

Τα σημαντικότερα Paper για το Buffer Overflow Exploit και τον εντοπισμό τους:

http://www.ultraviolet.org/mail-archives/bugtraq.2022/0553.html
Smashing The Stack For Fun And Profit

http://www.infosys.tuwien.ac.at/Staff/chris/
T.Toth, Christopher Kruegel, Ph.D.(PAPER)

Web Sites για την ασφάλεια υπολογιστικών συστημάτων:

http://www.sans.org/top20/
SANS Top 20 Vulnerabilities - The Experts Consensus</A>

http://www.metasploit.com/
www.metasploit.com

http://www.infosys.tuwien.ac.at/Staff/tt/abstract_execution/index.html
Detecting Buffer Overflow Exploits in Requests via Abstract Payload Execution

http://community.core-sdi.com/~juliano/protec.html
Protecting Against Buffer Overflows

http://www.stsc.hill.af.mil/crosstalk/2001/01/mchugh.html
STSC CrossTalk - Intrusion Detection Implementation and Operational Issues - Jan 2001

http://news.com.com/2100-1001-33483.html?legacy=cnet&tag=st.ne.1002.bgif%3fst.ne.fd.gif.l
Study says buffer overflow is most common security bug CNET News.com

http://www1.corest.com/common/showdoc.php?idx=221&idxseccion=10
DETECTION VULNERABILITIES

http://community.core-sdi.com/~juliano/
dc0ded Περιέχει σχεδόν όλα τα άρθρα των Hackers που χετίζονται με τα Buffer Overflow Exploits

http://www1.corest.com/common/showdoc.php?idx=221&idxseccion=10
CoreLabs -Advisories

http://www.computerworld.com/securitytopics/security/story/0,10801,84510,00.html
Blaster worm linked to severity of blackout - Computerworld

http://www.insecure.org/sploits_microshit.html
Exploit world -- Microsoft Windows, WindowsNT, Windows98, Windows95, and bloated programs secti

http://www.immunix.org/documentation.html Immunix.org
The Source for Secure Linux Components and Platforms

http://www.cse.ogi.edu/DISC/projects/immunix/StackGuard/usenixsc98_html/paper.html
StackGuard Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks

http://webster.cs.ucr.edu/Page_AoALinux/HTML/AoATOC.html The Art of Assembly
Language(THE BEST)

http://www.isacc.com/isacc2000/presentations/3c-vs/sld001.htm
Towards Certifying Software for Security

http://www.cee.hw.ac.uk/~alison/SysLevProg/t1/topic1.html
Your Compiled C Program Looking at the Assembly - NOTE

Θεωρία για τα λειτουργικά συστήματα

http://portal.acm.org/citation.cfm?id=173682.165159&coll=portal&dl=ACM&idx=J89&part=newsletter&WantType=newsletter&title=ACM%20SIGARCH%20Computer%20Architecture%20
News Citation

http://www.iecc.com/linker/linker10.html
Dynamic Linking and Loading

http://www.exposecorp.com/embedded/ex386.htm
Embedded 80386 Programming Examples-U

http://www.exposecorp.com/embedded/ex386.htm
Embedded 80386 Programming Examples

http://kernel.kaist.ac.kr/~jinsoo/course/cs330-2002spring/slides/supp-vm.pdf
http://www.amd.com/epd/processors/6.32bitproc/x21086/21086.pdf
http://www.netwinder.org/~scottb/notes/Elf-Design.html#toc4
NetWinder ELF Design Notes

http://www-106.ibm.com/developerworks/library/l-shobj/
Shared objects for the object disoriented

http://vx.netlux.org/texts/html/books/icz/tut4.html
Iczelion's Win32 Assembly Tutorial 4 Painting with Text

http://www.geocities.com/SiliconValley/Park/3230/x86asm/asmles00.html
Roby's PC Assembly Tutorial

Web Sites για τις μεθόδους επιθέσεων των Hackers

http://www.phrack.org/
PHRACK-MAGAZIN

http://packetstormsecurity.nl/
[packet storm]. - http--packetstormsecurity.org

http://www.garlic.com/~lynn/2002.html#23
2002 Newsgroup postings (1-1 - 1-12) Lynn Wheeler

http://www.cs.rice.edu/~scrosby/hash/
Algorithmic Complexity Attacks

http://webster.cs.ucr.edu/Page_AoALinux/0_AoAHLA.html
Art of Assembly Language Programming and HLA by Randall Hyde

http://www.asciitable.com/
Ascii Table - ASCII character codes and html, octal, hex and decimal charts

http://www.peterindia.com/AssemblyLanguage.html
Assembly Language Web Resources

http://www.iana.org/assignments/port-numbers
Assignments-PORT-Numbers

http://www.iro.umontreal.ca/~dift6221/bc_tutorial/tutorial/chapter4.htm
Behavioral Compiler Tutorial

http://www.is.titech.ac.jp/~wakita/classes/soft-2001/boflow-bib.pdf
BIBIOGRAFI REFf

http://www.isacc.com/isacc2000/presentations/3c-vs/sld026.htm
Buffer Overflow Detection

http://www.cs.unm.edu/~immsec/
Immune Systems - Main Page

http://gcc.gnu.org/
GCC Home Page - GNU Project - Free Software Foundation (FSF)

http://brand107.home.attbi.com/pc-gpe/intel.doc
http--brand107.home.attbi.com-pc-gpe-intel.doc

http://www-2.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15213-f02/www/R04/section_c/R04-sectionC-4up.pdf
http://www.11a.nu/stack/adv.overflow.paper.txt
http://www.cs.fsu.edu/~engelen/courses/COP402001/notes7_4.pdf
http://linux.oreillynet.com/pub/a/linux/2003/06/02/snort.html?page=2
SNORT HOLES [Jun. 02, 2003]

http://www.kb.cert.org/vuls/id/823260
SOS Vulnerability Note VU#823260 BUFFER OVERFLOW

ΕΥΧΑΡΙΣΤΙΕΣ

Αυτή η πτυχιακή εργασία που αφορά τα Buffer Overflow Exploits και τις μεθόδους εντοπισμού τους ειναι αποτέλεσμα της ενασχόλησης μου επί ένα χρόνο με τον τομέα της ασφάλειας τον υπολογιστικών συστημάτων. Η ενασχόληση μου με αυτό τον τομέα προέκυψε από το γενικότερο ενδιαφέρον μου, κατά την διάρκεια των σπουδών μου στο Τμήμα Πληροφορικής του ΤΕΙ Αθηνάς, στην ανάπτυξη εφαρμογών που μία από τις πιο κρίσιμες προδιαγραφές τους είναι η ασφάλεια. Την επιλογή μου αυτή την ενίσχυσε το γεγονός ότι εκπόνησα την πρακτική μου άσκηση στο εργαστήριο Ariadne-t του υπολογιστικού κέντρου του ΕΚΕΦΕ «Δημόκριτος».

Στην Ariadne-t είχα την τύχη να εμπλακώ σε μία διαδικασία έρευνας πάνω στην ασφάλεια που γίνεται σταθερά εκεί. Κατά την διάρκεια της πρακτικής μου άσκησης εκεί διδάχτηκα πώς η ασφάλεια αντιμετωπίζεται σε διάφορα επίπεδα από ένα Πληροφοριακό Σύστημα που βασίζεται σε δίκτυα, μέχρι και σε επίπεδο Υπολογιστικού Συστήματος ή Δικτυακών Υπηρεσιών. Η τάση μου να αντιμετωπίζω ένα πρόβλημα από την ρίζα του και από την σκοπιά του προγραμματιστή, με έκανε να διαλέξω την πρόταση πτυχιακής που αφορά το Buffer Overflow την όποια μου πρότεινε ο Χάρης Κουτσούρης ένα από τα βασικά μέλη του εργαστηρίου Ariadne-t. Δέχτηκα με χαρά να αναλάβω την εργασία που μου εμπιστεύθηκε το εργαστήριο αφού ήταν ένα από τα σημαντικότερα θέματα που απασχολούσε και απασχολεί την Ariadne-t μέχρι σήμερα.

Από την συνολική μου δουλειά κατά την διάρκεια της πτυχιακή μου εργασίας και της πρακτικής μου άσκησης στην Ariadne-t διαπίστωσα ότι ο μόνος τρόπος για να αντιμετωπιστούν αποτελεσματικά τα προβλήματα ασφάλειας είναι ο καλός σχεδιασμός ενός πληροφοριακού ή υπολογιστικού συστήματος ή δικτυακής υπηρεσίας πριν αυτή ενσωματωθεί σε παραγωγικό περιβάλλον. Επίσης διαπίστωσα ότι η ασφάλεια δεν πρέπει να είναι αυτοσκοπός όταν σχεδιάζεται ή αναπτύσσεται ένα πληροφοριακό σύστημα αλλά πρέπει να λαμβάνεται σοβαρά υπόψη όταν και στο βαθμό που αυτή χρειάζεται. Τέλος διαπίστωσα ότι η ασφάλεια στα υπολογιστικά συστήματα δεν μπορεί να εφαρμοστεί με την χρήση ενός και μόνο προϊόντος για παράδειγμα την κρυπτογραφία ή την λύση του προβλήματος των Buffer Overflow Exploit. Η υλοποίηση της όμως μπορεί να γίνει με το να εφαρμοστεί μια πολιτική ασφάλειας που η διάρκεια της θα ξεκινάει από την δημιουργία/ανάπτυξη του εκάστοτε πληροφοριακού συστήματος, ενώ η έκταση της θα αρχίζει από τις εφαρμογές(προγράμματα) και θα επεκτείνεται μέχρι τους ανθρώπους που συμμετέχουν σε αυτό.

Τελειώνοντας οφείλω να ευχαριστήσω τον Δρ. Ιωάννη Κοροβέση τον υπεύθυνο του εργαστηρίου Ariadne-t του ΕΚΕΦΕ «Δημόκριτος» που διέθεσε τους πόρους του εργαστηρίου για να μπορέσω να κάνω την πρακτική μου άσκηση και την πτυχιακή μου εργασία καθώς και για την στήριξη αυτής μου τις προσπάθειας. Φυσικά θέλω να ευχαριστήσω εξίσου τον εισηγητή καθηγητή της πτυχιακής μου Δρ. Ιωάννη Χάλαρη αφενός γιατί δέχτηκε να εισηγηθεί την πτυχιακή μου εργασία αφετέρου για τις πολύτιμες βάσεις που μου έδωσε πάνω στην ανάλυση και ανάπτυξη εφαρμογών μεγάλης κλίμακας μέσω των μαθημάτων του στο Τμήμα Πληροφορική του Τ.Ε.Ι Αθήνας. Φυσικά αισθάνομαι τυχερός που γνώρισα τα μέλη της Ariadne-t Χ. Κουτσούρη και Κ. Μάγκο που η εμπειρία τους και η γνώσης τους πάνω στα δίκτυα και την

ασφάλεια δικτύων ήταν ένα πολύτιμο εργαλείο για εμένα ώστε να μπορέσω να αναπτύξω ένα ικανοποιητικό υπόβαθρο και έτσι να χειρίζομαι με ευκολία θέματα όπως τα UNIX συστήματα το Debugging μιας δικτυακής τοπολογίας και τα NIDS. Τέλος δε πρέπει να παραλείψω τον Λευτέρη Γαριφαλίδη τον συμφοιτητή, συνάδελφο, και φίλο για την χρήσιμη βοήθεια του κάποιες στιγμές κατά την διάρκεια της πτυχιακής μου εργασίας.