Πανεπιστήμιο Πειραιώς
Τμήμα Ψηφιακών Συστημάτων

ΠΡΟΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ

«Ψηφιακών Συστημάτων»

«ΑΣΦΑΛΕΙΑ ΣΥΝΘΗΜΑΤΙΚΩΝ - ΕΝΑ ΠΡΟΓΡΑΜΜΑ ΑΝΑΚΤΗΣΗΣ ΚΩΔΙΚΩΝ ΜΕ ΕΠΙΘΕΣΕΙΣ ΤΥΠΟΥ ΛΕΞΙΚΟΥ»

Του φοιτητή: Κωστάντου Κων/νου - Ε/04076

Επιβλέπων καθηγητής: Κάτσικας Σ.

Πειραιάς 2009

Περιεχόμενα

Πρόλογος

Στο πρώτο μέρος η εργασία, αναφέρεται:

- στα συνθηματικά που χρησιμοποιούν οι χρήστες για να εξασφαλίσουν εμπιστευτικότητα σε ένα σύστημα
- στις βασικές αδυναμίες που υπάρχουν στα συνθηματικά
- σε τρόπους, ώστε οι χρήστες να δημιουργούν ισχυρά συνθηματικά και να είναι εύκολα στην απομνημόνευση
- στις μονόδρομες συναρτήσεις hash, που χρησιμοποιούνται για να εξασφαλίσουν υπηρεσίες ακεραιότητας και εμπιστευτικότητας.
- σε προγράμματα που υπάρχουν για ανάκτηση συνθηματικών

Στο δεύτερο μέρος της εργασίας παρουσιάζεται το πρόγραμμα dedalus, το οποίο είναι ένα πρόγραμμα που κάνει ανάκτηση κωδικών με τη χρήση επιθέσεων τύπου λεξικού.

Μερος 1ο

Καφάλαιο 1ο – Εισαγωγή και βασικές έννοιες

Εισαγωγή

Μέρος της ασφάλειας ενός πληροφοριακού συστήματος αποτελεί ο έλεγχος της ταυτότητας των χρηστών του. Αυτό γίνετε με την ταυτοποίηση και την αυθεντικοποίηση. Η ταυτοποίηση ενός λογικού υποκειμένου καλείτε η διαδικασία εκείνη κατά την οποία το λογικό υποκείμενο παρέχει σε ένα πληροφοριακό σύστημα τις πληροφορίες που απαιτούνται προκειμένου να συσχετιστεί με ένα από τα αντικείμενα που δικαιούνται προσπέλαση στους πόρους του. Αυθεντικοποίηση ενός λογικού υποκειμένου καλείτε η διαδικασία εκείνη κατά την οποία ένα λογικό υποκείμενο παρέχει σε ένα πληροφοριακό σύστημα τις πληροφορίες που απαιτούνται προκειμένου να ελεγχθεί η βασιμότητα της συσχέτισης που επιτεύχθηκε κατά τη διαδικασία της ταυτοποίησης. Η αυθεντικοποίηση αφορά επομένως την διαδικασία κατά την οποία επαληθεύετε η δηλωθείσα ταυτότητα ενός λογικού υποκειμένου [1].

Συνθηματικό είναι η πληροφορία η οποία σχετίζετε με ένα λογικό υποκείμενο και η οποία επιβεβαιώνει την ταυτότητα του λογικού υποκειμένου. Τα συνθηματικά αποτελούν το συνηθέστερο μέσο αυθεντικοποίησης. Τα συνθηματικά ανήκουν σε εκείνους τους μηχανισμούς αυθεντικοποίησης που βασίζονται σε κάτι που τα λογικά αντικείμενα (οι χρήστες) γνωρίζουν [1].
Το πρόβλημα που υπάρχει με τα συνθηματικά χρηστών περιγράφεται με τις παρακάτω προτάσεις [2]:

- Τα συνθηματικά πρέπει να είναι αρκετά ισχυρά, αλλά
- Τα συνθηματικά πρέπει να απομνημονεύονται από τους χρήστες.

Οι πολιτικές ασφαλείας για την ορθή χρήση των συνθηματικών απαιτούν από τους χρήστες να περιλαμβάνουν στα συνθηματικά τους κεφαλαία και πεζά γράμματα καθώς και αριθμούς όπως επίσης και ειδικούς χαρακτήρες πληκτρολόγησης. Όλα αυτά όμως έχουν ως αποτέλεσμα οι λέξεις που

χρησιμοποιούν οι χρήστες σαν συνθηματικά να μην έχουν πλέον κάποια φυσική σημασία και να γίνονται όλο και περισσότερο δύσκολες στην απομνημόνευση.

Η σύγχρονη κρυπτογραφία βασίζεται σε μονόδρομες συναρτήσεις, οι οποίες είναι εύκολο να υπολογιστούν αλλά δύσκολο να αναστραφούν. Οι μονόδρομες συναρτήσεις hash παράγουν στην έξοδο τους μηνύματα σταθερού μήκους ανεξάρτητα από το μήκος του μηνύματος εισόδου. Οι συναρτήσεις hash χρησιμοποιούνται κυρίως για να εξασφαλίσουν εμπιστευτικότητα στην ακεραιότητα δεδομένων και στην αυθεντικοποίηση μηνυμάτων.

Επιθέσεις λεξικού

Στην κρυπτανάλυση και στην ασφάλεια πληροφοριακών συστημάτων η επίθεση λεξικού είναι ένας κοινός τρόπος επίθεσης εναντίον στο hash ενός συνθηματικού. Ο επιτιθέμενος χρησιμοποιεί ένα αρχείο λεξικού και υπολογίζει την τιμή του hash κάθε λέξης. Συγκρίνει την τιμή, με την τιμή του hash του συνθηματικού, που επιθυμεί να ανακτήσει [3].

Επιθέσεις εξαντλητικής αναζήτησης

Στη κρυπτανάλυση ο όρος επίθεση εξαντλητικής αναζήτησης αναφέρετε στη μέθοδο όπου δοκιμάζονται όλοι οι δυνατοί συνδυασμοί από ένα εύρος χαρακτήρων με σκοπό να αποκρυπτογραφηθεί ένα μήνυμα. Θεωρητικά το ποσοστό επιτυχίας σε μία επίθεση εξαντλητικής αναζήτησης είναι εκατό τοις εκατό, αλλά όσο αυξάνετε το μέγεθος των συνθηματικών τόσο αυξάνονται και οι πιθανοί συνδυασμοί που θα δοκιμαστούν. Το πλήθος (Π) των πιθανών κωδικών που δοκιμάζονται δίνεται από τον παρακάτω τύπο:

$$\Pi=M^{\mu}$$

όπου Π είναι το πλήθος των συνδυασμών που θα δοκιμαστούν, Μ το πλήθος των χαρακτήρων που θα χρησιμοποιηθούν και μ το μήκος του αλφαριθμητικού που θέλουμε να δοκιμαστεί [2, 3].

Εντροπία

Η εντροπία είναι ένας όρος που αποδόθηκε από τον Claude Shannon και μετράει το πόση πληροφορία παράγεται από μία λέξη. Όταν αναφέρεται σε ένα κωδικό, ουσιαστικά δείχνει το πόσο εύκολα μπορεί κάποιος να «μαντέψει» τον κωδικό και συχνά αναφέρεται ο όρος «εντροπία μαντείας»[4, 2].

Κεφάλαιο 2ο – Κρυπτογραφικές συναρτήσεις HASH

ΚΡΥΠΤΟΓΡΑΦΙΚΕΣ ΣΥΝΑΡΤΗΣΕΙΣ HASH

Οι κρυπτογραφικές συναρτήσεις hash (cryptographic hash functions) κατέχουν σημαντικό ρόλο στην σύγχρονη κρυπτογραφία [5], χρησιμοποιούνται κυρίως για να εξασφαλίσουν εμπιστευτικότητα στην ακεραιότητα δεδομένων και στην αυθεντικοποίηση μηνυμάτων.

Οι συναρτήσεις hash δέχονται ένα μήνυμα σαν είσοδο και υπολογίζουν μία έξοδο γνωστή ως hash-code, hash-result, hash-value ή απλώς hash. Πιο συγκεκριμένα μία συνάρτηση hash δέχεται ως είσοδο μηνύματα μη συγκεκριμένου αλλά πεπερασμένου μήκους bits και υπολογίζει την έξοδο που είναι ένα μήνυμα σταθερού μήκους bits [4].

Ο πρώτος επίσημος ορισμός των μονόδρομων συναρτήσεων hash δόθηκε αρχικά από τους R. Merkle και M. Rabin

Μονόδρομη συνάρτηση hash είναι η συνάρτηση h που ικανοποιεί τις παρακάτω συνθήκες [6]:

1. Η περιγραφή της h πρέπει να είναι δημοσίως γνωστή και να μην χρειάζεται κάποια μυστική πληροφορία για την λειτουργία της.
2. Η μεταβλητή εισόδου της συνάρτησης μπορεί να είναι μη συγκεκριμένου μήκους αλλά η έξοδος της συνάρτησης πρέπει να είναι σταθερού μήκους n bits ($n \geq 64$).
3. Δοσμένης της συνάρτησης h και μεταβλητής X να είναι «εύκολο» υπολογίσιμο το h(X).
4. Η συνάρτηση hash πρέπει να είναι μονόδρομη, υπό την έννοια ότι δοθέντος Y το οποίο είναι η έξοδος μίας συνάρτησης h είναι «δύσκολο» να βρούμε X τέτοιο ώστε $h(X)=Y$ και δοθέντος X και h(X) είναι «δύσκολο» να βρεθεί X΄ τέτοιο ώστε για $X' \neq X$ να υπάρχει $h(X')=h(X)$.

Η ιδέα των μονόδρομων συναρτήσεων Hash (one-way hash function) εμφανίστηκε και χρησιμοποιήθηκε απευθείας για πρακτική εφαρμογή. Θεωρώντας την διαδικασία εισόδου (login) σε ένα υπολογιστικό σύστημα πολλών χρηστών (multiuser), κατά την δημιουργία του λογαριασμού χρήστη (user account), ο χρήστης επιλέγει ένα συνθηματικό (password) το οποίο αποθηκεύεται στο σύστημα σε ένα αρχείο που περιέχει τα συνθηματικά όλων των χρηστών. Κάθε φορά που ο χρήστης εισέρχεται στο σύστημα του ζητείται το συνθηματικό, το οποίο το σύστημα το συγκρίνει με το ήδη αποθηκευμένο. Οι αποθηκευμένοι κωδικοί πρέπει να φυλάσσονται μυστικοί έτσι ώστε να αποφύγουμε την είσοδο ενός κακόβουλου χρήστη στο σύστημα [6].

Ο Needham [7] πρώτος διαπίστωσε ότι είναι εφικτό ένα σύστημα να κρίνει την αυθεντικότητα ενός συνθηματικού χωρίς το σύστημα να γνωρίζει το συνθηματικό. Όταν ο χρήστης εισάγει για πρώτη φορά ένα συνθηματικό στο σύστημα , κατά τη διαδικασία δημιουργίας λογαριασμού χρήστη, το σύστημα υπολογίζει το αποτέλεσμα μιας συνάρτησης h(pw) (όπου pw το συνθηματικό του χρήστη) και αποθηκεύει αυτό αντί για το συνθηματικό αυτούσιο. Κάθε φορά που ένας χρήστης χρησιμοποιεί ένα συνθηματικό X για την είσοδο στο σύστημα, το σύστημα συγκρίνει το h(pw) με το h(X) και επιτρέπει την είσοδο στο χρήστη εάν αυτά τα δύο είναι ιδία. Η όλη επιτυχία του συστήματος βασίζονταν στο εάν η συνάρτηση h είναι μονόδρομη ή όχι, δηλαδή για οποιαδήποτε είσοδο να είναι εύκολο να υπολογιστεί το αποτέλεσμα άλλα αδύνατο να αντιστραφεί, ακόμα και ένα ήταν γνωστή η συνάρτηση h και το αποτέλεσμα h(pw) κάποιος κακόβουλος χρήστης να μην μπορεί να υπολογίσει το συνθηματικό.

MD5

Ο MD5 είναι ένας αλγόριθμος hash που δέχεται ως είσοδο ένα μήνυμα μεγέθους M και έχει ως έξοδο μία τιμή hash μήκους 128 bits. Ο MD5 ήταν ο τελευταίος επιτυχημένος αλγόριθμος κρυπτογράφησης hash που σχεδιάστηκε από τον Ron Rivest το 1992. Είναι πολύ γνωστός και χρησιμοποιείτε ευρέως σε πολλές εφαρμογές περιλαμβάνοντας SSL/TLS και IPsec και πολλά άλλα κρυπτογραφικά πρωτόκολλα [8, 9].

SHA - secure hash standard

Το πρότυπο sha περιλαμβάνει τέσσερις αλγορίθμους hash, sha-1, sha-256, sha-384, sha-512. Και οι τέσσερις αλγόριθμοι είναι μονόδρομες συναρτήσεις hash που μπορούν να επεξεργαστούν ένα μήνυμα και να παράγουν μία σύνοψη του μηνύματος που ονομάζεται Message Digest . Καθένας από τους παραπάνω αλγόριθμους περιγράφεται με δύο βήματα, την προ-επεξεργασία και τον υπολογισμό του hash. Η διαφορά των τεσσάρων αλγορίθμων είναι κυρίως ο αριθμός των bits ασφαλείας που παρέχουν στα δεδομένα που περνάνε από την συνάρτηση hash, το οποίο έχει άμεσο αποτέλεσμα με το μέγεθος του hash που παράγεται. Ακόμη οι τέσσερις αλγόριθμοι διαφέρουν στο μέγεθος των blocks και το μήκος λέξεων που χρησιμοποιούνται κατά την διαδικασία υπολογισμού του hash. Ο πίνακας 1 παρουσιάζει τις βασικές ιδιότητες των τεσσάρων αλγορίθμων του προτύπου sha [10].

Algorithm	Message Size (bits)	Block Size (bits)	Word Size (bits)	Message Digest Size (bits)	Security (bits)
SHA-1	$< 2^{64}$	512	32	160	80
SHA-256	$< 2^{64}$	512	32	256	128
SHA-384	$< 2^{128}$	1024	64	384	192
SHA-512	$< 2^{128}$	1024	64	512	256

Πίνακας 1. Ιδιότητες των αλγορίθμων του προτύπου sha.

Κεφάλαιο 3° - Αδυναμίες στα συνθηματικά

Όλα τα συστήματα αυθεντικοποίησης υποφέρουν από ένα βασικό πρόβλημα, την περιορισμένη ανθρώπινη μνήμη. Εάν οι άνθρωποι δεν χρειαζόταν να απομνημονεύουν τα συνθηματικά τότε το συνθηματικό με την μέγιστη ασφάλεια θα ήταν αυτό με την μέγιστη εντροπία: θα αποτελούταν από ένα αλφαριθμητικό με το μέγιστο μέγεθος που επιτρέπει το σύστημα, χρησιμοποιώντας όλους τους χαρακτήρες που επιτρέπει το σύστημα και με εντελώς τυχαία σειρά διαδοχής ο ένας χαρακτήρας από τον άλλο [11]. Οι περισσότεροι μεγάλοι οργανισμοί συμβουλεύουν τους χρήστες να χρησιμοποιούν έναν «καλό κωδικό». Ένας «καλός κωδικός» με τον όρο που χρησιμοποιούν, θα πρέπει να είναι τυπικά μεγάλος, τουλάχιστον οκτώ χαρακτήρων, και να χρησιμοποιεί ένα μεγάλο εύρος διαφορετικών χαρακτήρων, όπως ειδικούς χαρακτήρες πληκτρολόγησης (πχ !@#&*) όπως επίσης και κεφαλαία και μικρά γράμματα [12], παρά όλα αυτά όμως πρέπει να απομνημονεύεται εύκολα. Για να αποφύγουν το να θυμούνται πολλά συνθηματικά οι χρήστες χρησιμοποιούν τον ίδιο κωδικό σε πολλά συστήματα και εφαρμογές. Έτσι ένας κωδικός εάν ανακτηθεί μπορεί να ανοίξει πολλές «πόρτες». Οι μεγάλοι κωδικοί που περιέχουν πλήθος διαφορετικών χαρακτήρων είναι δύσκολο να ανακτηθούν αλλά και πολύ δύσκολο να απομνημονευτούν, έτσι οι χρήστες οδηγούνται στο να γράφουν τους κωδικούς σε σημεία με πρόσβαση από πολύ κόσμο [11].

Για να απομνημονεύουν εύκολα τους κωδικούς τους οι χρήστες μπορούν να χρησιμοποιούν μία MPF (Mnemonic Password Formula). Η MPF είναι μία τεχνική όπου η χρήστες μπορούν να κατασκευάζουν ένα ισχυρό συνθηματικό από διάφορες πληροφορίες που έχουν συνεχώς πρόσβαση σε αυτές, έτσι ώστε να είναι και εύκολα απομνημονευμένο [13].

Ένας κωδικό που δημιουργείται από μία καλά σχεδιασμένη MPF πρέπει να έχει τις παρακάτω ιδιότητες:

1. Να είναι όσο το δυνατών περισσότερο ένα αλφαριθμητικό από τυχαία ακολουθία χαρακτήρων.

2. Να είναι μεγάλο σε μέγεθος και αρκετά πολύπλοκο, έτσι ώστε να είναι δύσκολο να ανακτηθεί από επιθέσεις εξαντλητικής αναζήτησης (brute force attacks).

3. Να είναι εύκολο να αναπαραχθεί από ένα χρήστη, όταν αυτός γνωρίζει μόνο την MPF και το σύστημα στο οποίο χρησιμοποιείται το συνθηματικό.

4. Να είναι μοναδικό για κάθε χρήστη.

Μία απλή MPF

Η ακόλουθη απλή MPF χρησιμοποιεί για να κατασκευάσουμε ένα συνθηματικό το όνομα χρήστη μίας εφαρμογής καθώς επίσης το όνομα του εξυπερετητή της εφαρμογής ή την τελευταία οκτάδα της IP διεύθυνσης του.

<username>!<hostname | lastoctet>

Η MPF θα παράγει τους ακόλουθους κωδικούς:

- "kosta!dtps" Για τον χρήστη kosta στο dtps.unipi.gr
- "kosta!51" Για τον χρήστη kosta στο σύστημα με IP 10.0.0.51

Αυτή η απλή MPF δημιουργεί ένα κωδικό με ικανοποιητικό μέγεθος, που περιέχει και έναν ειδικό χαρακτήρα. Παρά όλα αυτά μπορούν να χρησιμοποιηθούν MPF πιο πολύπλοκες για μεγαλύτερο μήκος συνθηματικών και με περισσότερους ειδικούς χαρακτήρες [13].

Κεφάλαιο 4° - Προγράμματα ανάκτησης συνθηματικών

Παρακάτω παρουσιάζονται πέντε βασικά προγράμματα ανάκτησης συνθηματικών για λειτουργικά συστήματα windows και Linux.

1. Cain and abel: Το cain and abel είναι ένα δωρεάν πρόγραμμα ανάκτησης κωδικών για λειτουργικά συστήματα windows. Επιτρέπει την ανάκτηση κωδικών με επιθέσεις εξαντλητικής αναζήτησης, επιθέσεις λεξικού καθώς και με υποκλοπή πακέτων από το δίκτυο[14,15].

2. John the ripper: Το John the ripper είναι ένα γρήγορο πρόγραμμα ανάκτησης κωδικών για λειτουργικά συστήματα windows και Linux. Κύριο σκοπό είχε να ανακτά αδύναμα συνθηματικά σε συστήματα Unix. Υποστηρίζει τις κρυπτογραφικέ συναρτήσεις hash όλων των διανομών των λειτουργικών συστημάτων Unix καθώς επίσης και του πρωτοκόλλου Kerberos AFS και των εκδόσεων του λειτουργικού συστήματος windows[14,15].

3. THC Hydra: Το thc hydra είναι ένα γρήγορο πρόγραμμα απομακρυσμένης αυθεντικοποίησης το οποίο τρέχει πάνω σε διάφορα πρωτόκολλα όπως http, ftp, telnet, smb, IMAP, pop3, VNC, Mysql και αρκετά άλλα. Μπορεί να κάνει επιθέσεις λεξικού και εξαντλητικής αναζήτησης [14,16].

4. L0phtCrack: Το L0phtCrack είναι ένα πρόγραμμα ανάκτησης συνθηματικών για λειτουργικά συστήματα windows μπορεί να κάνει ανάκτηση συνθηματικών χρησιμοποιώντας επιθέσεις εξαντλητικής αναζήτησης και λεξικού [14,15].

5. RainbowCrack: Το RainbowCrack είναι ένα πρόγραμμα εύρεσης των αρχικών τιμών μίας τιμής hash (hash cracker). Το RainbowCrack χρησιμοποιεί μία σχέση μεταξύ χρόνου και μνήμης, χρειάζεται αρκετή ώρα να παράγει όλες τις πιθανές τιμές hash τις οποίες αποθηκεύει σε πίνακες που ονομάζονται RainbowTables, αλλά μόλις δημιουργηθούν

οι πίνακες μπορεί να τους διαβάζει πολύ γρήγορα για να ανακτήσει την αρχική τιμή του hash [14,17].

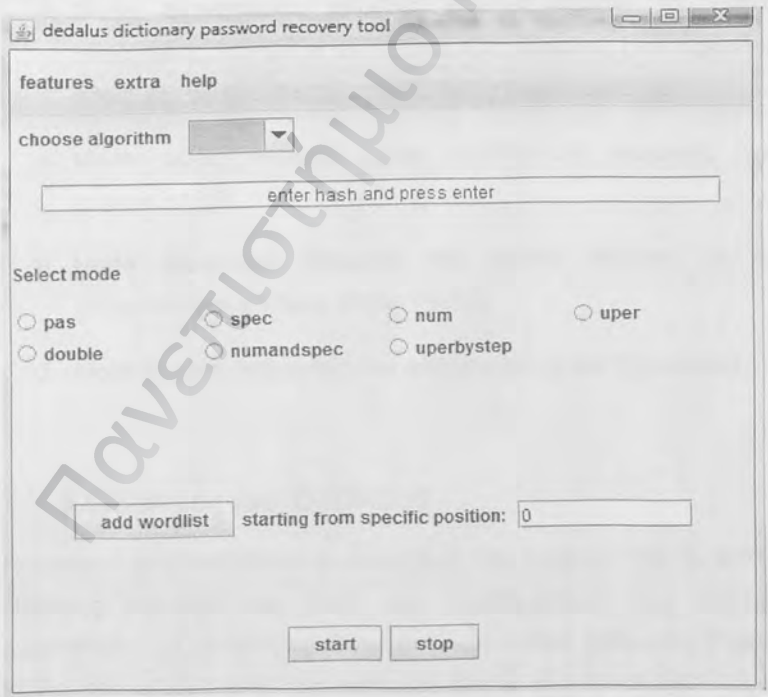
Συμπεράσματα

Οι συναρτήσεις hash δέχονται ως είσοδο ένα με συγκεκριμένου αλλά πεπερασμένου μήκους μήνυμα και υπολογίζουν ως έξοδο την τιμή hash που είναι ένα μήνυμα σταθερού μήκους. Πρώτος ο Needham σχεδίασε ένα σύστημα όπου εξασφαλίζει εμπιστευτικότητα στην είσοδο των χρηστών χωρίς το σύστημα να γνωρίζει το συνθηματικό του χρήστη. Ο έλεγχος ταυτότητας ενός χρήστη σε ένα πληροφοριακό σύστημα γίνετε με την ταυτοποίηση και την αυθεντικοποίηση. Η αυθεντικοποίηση είναι η διαδικασία κατά την οποία επαληθεύετε η δηλωθείσα ταυτότητα του χρήστη. Τα συνθηματικά αποτελούν το συνηθέστερο μέσο αυθεντικοποίησης. Οι κωδικοί που χρησιμοποιούν συνήθως οι χρήστες υποφέρουν κυρίως από τρείς βασικές αδυναμίες. Είναι εύκολα συνθηματικά με μικρή εντοπία που μπορούν να ανακτηθούν εύκολα από ένα πρόγραμμα που κάνει επιθέσεις εξαντλητικής αναζήτησης ή να ανακτηθούν από ένα πρόγραμμα που κάνει επιθέσεις τύπου λεξικού. Χρησιμοποιούν ένα συνθηματικό για πολλές εφαρμογές, οπότε εάν ανακτηθεί δίνει πρόσβαση σε πολλά σημεία και γράφουν τα συνθηματικά σε μέρη με πρόσβαση από πολύ κόσμο. Για να απομνημονεύουν εύκολα ισχυρά συνθηματικά οι χρήστες μπορούν να χρησιμοποιούν μία MPF (Mnemonic Password Formula)

Μερος 2ο

Εισαγωγή

Οι ανάκτηση των συνθηματικών είναι αναγκαία όταν ένας χρήστης ενός συστήματος δεν είναι πλέον σε θέση να αυθεντικοποιηθεί λόγω του ότι έχει χάσει ή ξεχάσει τον κωδικό του [18]. Το dedalus είναι ένα πρόγραμμα ανάκτησης κωδικών με τη χρήση επιθέσεων λεξικού για κωδικούς που έχουν κρυπτογραφηθεί με md5, sha-1, sha-256, sha-384 και sha-512. Επίσης περιέχει ένα υπολογιστή τιμών hash (hash calculator) για να υπολογίζουν οι χρήστες τα hash που παράγονται από τους κωδικούς τους και να μπορούν να τα δοκιμάζουν. Το dedalus έχει υλοποιηθεί σε γλώσσα προγραμματισμού Java και είναι σε εκτελέσιμο αρχείο .jar έτσι ώστε να τρέχει σε λειτουργικά συστήματα windows και linux.



Εικόνα 1. Αρχική οθόνη του dedalus.

Δυνατότητες επίθεσης

Το dedalus έχει επτά δυνατότητες (modes) επίθεσης βασισμένες σε επιθέσεις λεξικού και υβριδικές επιθέσεις λεξικού και εξαντλητικής αναζήτησης

1. Mode pas: δοκιμάζει τον κωδικό όπως το διαβάζει από το λεξικό (pass→pass).
2. Mode num: υβριδική επίθεση λεξικού και εξαντλητικής αναζήτησης. Δοκιμάζει τον κωδικό προσθέτοντας δύο αριθμούς στο τέλος κάθε λέξεις από 00…99 (pass→pass00…pass99).
3. Mode spec: υβριδική επίθεση λεξικού και εξαντλητικής αναζήτησης. Δοκιμάζει τον κωδικό προσθέτοντας δύο ειδικούς χαρακτήρες στο τέλος κάθε λέξεις (pass→pass@#).
4. Mode numandspec: υβριδική επίθεση λεξικού και εξαντλητικής αναζήτησης. Δοκιμάζει τον κωδικό προσθέτοντας έναν αριθμό και έναν ειδικό χαρακτήρα στο τέλος κάθε λέξης (pass→pass1@).
5. Mode uper: δοκιμάζει στον κωδικό σε κεφαλαία γράμματα (pass→PASS).
6. Mode uperbystep: δοκιμάζει τον κωδικό σταδιακά σε κεφαλαία γράμματα (pass→Pass..PaSs..PASS).
7. Mode double: διπλασιάζει τον κάθε κωδικό (pass→passpass).

Αλγόριθμος του DEDALUS

Παρακάτω παρουσιάζονται οι αλγόριθμοι που χρησιμοποιεί το dedalus στις επιθέσεις εναντίον των hash των συνθηματικών που πρόκειται να ανακτηθούν. Οι μεταβλητές αναφέρονται με *πλάγια γράμματα*. Η συνάρτηση EOF() επιστρέφει 1 όταν έχει διαβαστεί όλο το αρχείο λεξικού, η συνάρτηση σύγκριση (αλφαριθμητικό, αλφαριθμητικό) επιστρέφει 0 ένα τα δύο

αλφαριθμητικά είναι ίδια, η συνάρτηση hash(αλφαριθμητικό) υπολογίζει την τιμή του hash του εκάστοτε αλφαριθμιτικού, η συνάρτηση γράμμα_στη_θέση(αλφαριθμητικό, αριθμός) επιστρέφει το γράμμα που βρίσκετε στην συγκεκριμένη θέση του αλφαριθμητικού, η συνάρτηση μήκος(αλφαριθμητικό) επιστρέφει το μήκος του αλφαριθμητικού και η συνάρτηση σε_κεφαλαία(αλφαριθμητικό) επιστρέφει το αλφαριθμητικό σε κεφαλαία γράμματα.

1) **Αλγόριθμος** modepas

Άνοιξε "wordlist.txt" για είσοδο ως #1

Όσο όχι EOF(1) **επανάλαβε**

Δίαβασε#1 *σειρά_κειμένου*

για_δοκιμή ← *σειρά_κειμένου*

Εάν σύγκριση(hash(*για_δοκιμή*),*κωδικός*)==0

Εκτύπωσε *για_δοκιμή*

Τέλος_επανάληψης

Κλείσε#1

Τέλος modepass

2) **Αλγόριθμος** modenum

Άνοιξε "wordlist.txt" για είσοδο ως #1

Όσο όχι EOF(1) **επανάλαβε**

Δίαβασε#1 *σειρά_κειμένου*

Για δείκτης1 **από** 0 **μέχρι** 9 [**με_βήμα** 1]

Για δείκτης2 **από** 0 **μέχρι** 9) [**με_βήμα** 1]

για_δοκιμή ← *σειρά_κειμένου*+ δείκτης1 + δείκτης2

Εάν σύγκριση(hash(*για_δοκιμή*),*κωδικός*)==0

Εκτύπωσε *για_δοκιμή*

Τέλος_επανάληψης

Κλείσε#1

Τέλος modenum

3) **Αλγόριθμος** modespec

Αλφαριθμητικό *spec*="!@#$%^&*"

Άνοιξε "wordlist.txt" για είσοδο ως #1

Όσο όχι EOF(1) **επανάλαβε**

Δίαβασε#1 *σειρά_κειμένου*

Για δείκτης1 **από** 0 **μέχρι** μήκος(*spec*) **[με_βήμα** 1]

Για δείκτης2 **από** 0 **μέχρι** μήκος(*spec*) **[με_βήμα** 1]

για_δοκιμή ← *σειρά_κειμένου*+ γράμμα_στη_θέση(*spec*,δείκτης1) + γράμμα_στη_θέση(*spec*,δείκτης2)

Εάν σύγκριση(hash(*για_δοκιμή*),*κωδικός*)==0

Εκτύπωσε *για_δοκιμή*

Τέλος_επανάληψης

Κλείσε#1

Τέλος modespec

4) **Αλγόριθμος** modenumandspec

Αλφαριθμητικό *numspec*="!@#$%^&*1234567890"

Άνοιξε "wordlist.txt" για είσοδο ως #1

Όσο όχι EOF(1) **επανάλαβε**

Δίαβασε#1 *σειρά_κειμένου*

Για δείκτης1 **από** 0 **μέχρι** μήκος(*numspec*) **[με_βήμα** 1]

Για δείκτης2 **από** 0 **μέχρι** μήκος(*numspec*) [**με_βήμα** 1]

για_δοκιμή ← *σειρά_κειμένου*+ γράμμα_στη_θέση(*numspec*,δείκτης1) + γράμμα_στη_θέση(*numspec*,δείκτης2)

Εάν σύγκριση(hash(*για_δοκιμή*),κωδικός)==0

Εκτύπωσε *για_δοκιμή*

Τέλος_επανάληψης

Κλείσε#1

Τέλος modenumandspec

5) **Αλγόριθμος** moduper

Άνοιξε "wordlist.txt" για είσοδο ως #1

Όσο όχι EOF(1) **επανάλαβε**

Δίαβασε#1 *σειρά_κειμένου*

για_δοκιμή ← σε_κεφαλαία(*σειρά_κειμένου*)

Εάν σύγκριση(hash(*για_δοκιμή*),*κωδικός*)==0

Εκτύπωσε *για_δοκιμή*

Τέλος_επανάληψης

Κλείσε#1

Τέλος modeuper

6) **Αλγόριθμος** moduperbystep

Άνοιξε "wordlist.txt" για είσοδο ως #1

Όσο όχι EOF(1) **επανάλαβε**

Δίαβασε#1 *σειρά_κειμένου*

Εάν μήκος(*σειρά_κειμένου*)==1

Κεφαλαία←σε_κεφαλαία(*σειρά_κειμένου*)

Βοηθιτικό←γράμμα_στη_θέση(*σειρά_κειμένου*,0)+ γράμμα_στη_θέση(*κεφαλαία*,0)

Για δείκτης1 **από** 0 **μέχρι** 1 [**με_βήμα** 1]

για_δοκιμή ← γράμμα_στη_θέση(*βοηθιτικό*,δείκτης1)

Εάν σύγκριση(hash(*για_δοκιμή*),*κωδικός*)==0

Εκτύπωσε *για_δοκιμή*

Τέλος_επανάληψης

Εάν μήκος(*σειρά_κειμένου*)==2

Κεφαλαία←σε_κεφαλαία(*σειρά_κειμένου*)

Βοηθιτικό1←γράμμα_στη_θέση(*σειρά_κειμένου,0*)+ γράμμα_στη_θέση(*κεφαλαία*,0)

Βοηθιτικό2←γράμμα_στη_θέση(σειρά_κειμένου,1)+ γράμμα_στη_θέση(*κεφαλαία*,1)

Για δείκτης1 **από** 0 **μέχρι** 1 [**με_βήμα** 1]

Για δείκτης2 **από** 0 **μέχρι** 1 [**με_βήμα** 1]

για_δοκιμή ← γράμμα_στη_θέση(*βοηθιτικό1*,δείκτης1)+ γράμμα_στη_θέση(*βοηθιτικό2*,δείκτης2)

Εάν σύγκριση(hash(*για_δοκιμή*),*κωδικός*)==0

Εκτύπωσε *για_δοκιμή*

Τέλος_επανάληψης

Εάν μήκος(*σειρά_κειμένου*)==3

Κεφαλαία←σε_κεφαλαία(*σειρά_κειμένου*)

Βοηθιτικό1←γράμμα_στη_θέση(*σειρά_κειμένου,0*)+ γράμμα_στη_θέση(*κεφαλαία*,0)

Βοηθιτικό2←γράμμα_στη_θέση(σειρά_κειμένου,1)+
γράμμα_στη_θέση(*κεφαλαία*,1)

Βοηθιτικό3←γράμμα_στη_θέση(σειρά_κειμένου,3)+
γράμμα_στη_θέση(*κεφαλαία*,3)

Για δείκτης1 **από** 0 **μέχρι** 1 [**με_βήμα** 1]

Για δείκτης2 **από** 0 **μέχρι** 1 [**με_βήμα** 1]

Για δείκτης3 **από** 0 **μέχρι** 1 [**με_βήμα** 1]

για_δοκιμή ← γράμμα_στη_θέση(*βοηθιτικό1*,δείκτης1)+
γράμμα_στη_θέση(*βοηθιτικό2*,δείκτης2)+
γράμμα_στη_θέση(*βοηθιτικό3*,δείκτης3)

Εάν σύγκριση(hash(*για_δοκιμή*),*κωδικός*)==0

Εκτύπωσε *για_δοκιμή*

Τέλος_επανάληψης

...

...

(Ο αλγόριθμος της δυνατότητας επίθεσης uperbystep συνεχίζετε καθ' αυτόν τον τρόπο για λέξης μήκους μέχρι δεκαπέντε (15) χαρακτήρων.)

Κλείσε#1

Τέλος modeuperbystep

7) **Αλγόριθμος** modedouble

Άνοιξε "wordlist.txt" για είσοδο ως #1

Όσο όχι EOF(1) **επανάλαβε**

Δίαβασε#1 *σειρά_κειμένου*

για_δοκιμή ← *σειρά_κειμένου*+*σειρά_κειμένου*

Εάν σύγκριση(hash(*για_δοκιμή*),*κωδικός*)==0

Εκτύπωσε *για_δοκιμή*

Τέλος_επανάληψης

Κλείσε#1

Τέλος modedouble

Δοκιμές σε συνθηματικά

Οι δοκιμές έγιναν σε υπολογιστή με λειτουργικό windows vista 32-bit, επεξεργαστή Intel® Core™2 DUO CPU T5250 @ 1.50GHz και μνήμη RAM 2046 MB. Το αρχείο λεξικού που χρησιμοποιήθηκε είχε μέγεθος 3.29MB και 306706 λέξεις.

Στον παρακάτω πίνακα ενδεικτικά φαίνονται οι χρόνοι που έκαναν να ανακτηθούν κωδικοί βασισμένοι σε λέξεις που μπορούν να βρεθούν στην αρχή, μέση και τέλος ενός αρχείου λεξικού. Η σύγκριση έγινε μεταξύ του προγράμματος dedalus και του cain and abel.

password	Hash function	Cracked from dedalus	Dedalus time	Cracked from cain	Cain time
druid	Md5	yes	1 sec	yes	1 sec
druid09	Sha-1	yes	75 sec	yes	13 sec
DRUID	Sha-512	yes	4 sec	yes	2 sec
druiddruid	Sha-256	yes	2 sec	yes	2 sec
AbEl	Md5	yes	5 sec	yes	3 sec
Abel#2	Sha-384	yes	4 sec	no	-
zythum$%	Md5	Yes	108 sec	no	-

Συμπεράσματα

Η επιτυχής ανάκτηση ενός κωδικού από το πρόγραμμα dedalus οφείλεται κυρίως στο αρχείο λεξικού που χρησιμοποιείται. Ένα μεγάλο αρχείο λεξικού με πολλές λέξεις και χρησιμοποιώντας την κατάλληλη δυνατότητα επίθεσης (mode) κάθε φορά έχει περισσότερες πιθανότητες να ανακτηθεί ο κωδικός. Ο χρόνος ανάκτησης οφείλεται στις δυνατότητες του συστήματος.

Βιβλιογραφία

[1] Σωκράτης Κάτσικας, Δημήτρης Γκριτζάλης, Στέφανος Γκριτζάλης. Ασφάλεια πληροφοριακών συστημάτων (2004).

[2] Saranga Komanduri. IMPROVING PASSWORD USABILITY WITH VISUAL TECHNIQUES (2007).

[3] Massimiliano Montoro. Cain & Abel - User Manual Copyright © 2001-2009

[4] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone. HANDBOOK of APPLIED CRYPTOGRAPHY, (1996).

[5] Diffie W, Hellman M. New directions in cryptography (1976).

[6] Pappu Srinivasa Ravikanth. Physical one-way functions (2001).

[7] Needham R, in Wilkes, M.V. Time-sharing computer systems (1972).

[8] Burt Kaliski, Matt Robshaw. Message Authentication with MD5 (1995).

[9] Bart PRENEEL. Analysis and Design of Cryptographic Hash Functions (2003).

[10] NIST. SECURE HASH STANDARD (2002).

[11] Jianxin Yan, Alan Blackwell, Ross Anderson, Alasdair Grant. The Memorability and Security of Passwords -Some Empirical Results (2001).

[12] Bonnie Chantarotwong. Password Security, (2003).

[13] Druid, C^2ISSP. Mnemonic Password Formulas (2006).

[14] top 10 password crackers. www.insecure.org (είσοδος 23/08/2009).

[15] Eric Cole. 40 Top Security Tools (2006).

[16] THC Hydra release. http://freeworld.thc.org (είσοδος 24/08/2009)

[17] RainbowCrack Tutorial. http://project-rainbowcrack.com (είσοδος 24/08/2009)

[18] Charles Miller. Password Recovery (2002).

.

.