Assignment 6: Systems security: Buffer overrun attack

Σκοπός
Ο στόχος της άσκησης αυτής είναι να βοηθήσει στην εκμάθηση/κατανόηση (1) της αναπαράστασης προγραμμάτων στη μνήμη, (2) της μορφής και χρήσης της στοίβας, (3) προβλημάτων ασφαλείας που υπάρχουν σε σημερινά συστήματα και προγράμματα.

Background
Η άσκηση αυτή θα γίνει σε **Linux/x86** συστήματα του τμήματος (portokali, milo, rodakino, etc.). Για θέματα της αρχιτεκτονικής του x86 και της γλώσσας assembly που χρησιμοποιεί μπορείτε να συμβουλευτείτε την online έκδοση του βιβλίου: Programming from the Ground Up, Jonathan Bartlett 2004. Ιδιαίτερα χρήσιμα θα βρείτε τα Chapter 3 και Appendix B.
Μέσα στο repo της άσκησης υπάρχει το README.md αρχείο όπου εκεί θα συμπληρώνετε τις απαντήσεις σας στα ερωτήματα.

Μέσα στο repo κάτω από τον φάκελο src θα βρειτε το πρόγραμμα hello.c και bufsize.h. Παρακαλείστε να κατεβάσετε μέσα και να το τοποθετήσετε μέσα στο src φάκελο το executable hello που βρίσκεται στην περιοχή του μαθήματος, και έχει δημιουργηθεί από το hello.c με την εντολή gcc και τις επιπλέον παραμέτρους "-static -g". Το πρόγραμμα ζητάει το όνομα σας και τυπώνει κάτι όπως το παρακάτω:

```
% hello
What is your name?
John
Thank you, John.
I recommend that you get a grade of 3 on this assignment.
```

Ωστόσο, ο συγγραφέας του προγράμματος έκανε το λάθος να μην ελέγχει τα όρια του array μέσα στο οποίο γίνεται η ανάγνωση του input κάνοντας το πρόγραμμα ευάλωτο σε επιθέσεις χρηστών.

Η άσκηση
Στην άσκηση αυτή πρέπει να προετοιμάσετε κατάλληλο "input" για το πρόγραμμα, ώστε να εκμεταλλεύεται το πρόβλημα αυτό και να τυπώνει:

```
% hello < data
What is your name?
Thank you, John.
I recommend that you get a grade of 9 on this assignment.
```

Όπως μπορείτε να δείτε από τον source κώδικα του προγράμματος, έχει γραφτεί ώστε να μην δίνει 9 σε κανένα, κάτω υπό οποιαδήποτε συνθήκες. Ωστόσο, ο κώδικας είναι γραμμένος με κακό τρόπο: διαβάζει input σε ένα buffer, αλλά δεν ελέγχει αν το input χωράει στον buffer. Αυτό σημαίνει ότι input το οποίο είναι αρκετά μεγάλο σε μήκος μπορεί να κάνει overwrite άλλες σημαντικές πληροφορίες στη μνήμη του προγράμματος, και να ξεγελάσει το πρόγραμμα ώστε να δώσει βαθμό 9 σε όποιονθέλετε.

1. Απαντήστε στην παρακάτω ερώτηση
Αναφέρετε στο readme file της άσκησης σας ποιο ή ποια από τα πρώτα 8 άρθρα (articles) της Συνθήκης για το Ηλεκτρονικό Έγκλημα (Convention on Cybercrime) καθιστούν παράνομη τη χρήση επίθεσης με buffer overrun attack; Θα χρειαστεί να διαβάσετε τα αντίστοιχα άρθρα της συνθήκης και ίσως βρείτε χρήσιμη και την σχετική αναφορά με επεξηγήσεις.

2. Αναλύστε το πρόγραμμα
Πάρτε το executable hello που δίνεται και χρησιμοποιήστε τον gdb για να αναλύσετε τα διάφορα μέρη του:

- Αναλύστε το **text** section χρησιμοποιώντας την εντολή "x":

```
% gdb hello
(gdb) x/79i readString
```

Αντιγράψτε τις 79 γραμμές του text section σε ένα αρχείο traces και προσθέστε σχόλια στον κώδικα που να εξηγούν τι συμβαίνει. Θα πρέπει να χρησιμοποιήσετε ως αναφορά τον source κώδικα του hello.c. Μάλιστα τα σχόλιά σας μπορεί να είναι απλά να δείξετε ποιες εντολές assembly αντιστοιχούν σε ποια κομμάτια του C κώδικα. Δεν χρειάζεται να εξηγήσετε τι κάνει η κάθε γραμμή assembly κώδικα χωριστά.

- Αναλύστε το **data** section με τις παρακάτω "print" εντολές:

```
% gdb hello
(gdb) print &grade
(gdb) print grade
```

Τοποθετήστε ένα διάγραμμα του data section στο αρχείο traces δείχνοντας με ακρίβεια τις μεταβλητές που περιέχει και τις διευθύνσεις τους.

- Αναλύστε το **bss** section με τις παρακάτω "print" εντολές:

```
% gdb hello
(gdb) print &Name
```

Τοποθετήστε ένα διάγραμμα του bss section στο αρχείο traces δείχνοντας με ακρίβεια τις μεταβλητές που περιέχει και τις διευθύνσεις τους.

- Αναλύστε το **stack-frame** της συνάρτησης readString. Το καλύτερο είναι να εξετάσετε το stack frame αφού η συνάρτηση έχει αρχίσει να εκτελείται και αφού έχει διαβάσει ένα όνομα στη μεταβλητή buf. Χρησιμοποιήστε τις παρακάτω εντολές για αυτό το σκοπό:

```
% gdb hello
(gdb) break *readString+73
(gdb) run
```

Δώστε ως input ένα όνομα

```
(gdb) print $esp
(gdb) print $ebp
(gdb) x/??b $esp (όπου ?? είναι ο κατάλληλος αριθμός από bytes)
```

Τοποθετήστε ένα διάγραμμα του stack frame της readString στο αρχείο traces δείχνοντας με ακρίβεια τις τοπικές μεταβλητές που περιέχει, τις input παραμέτρους, και τη διεύθυνση επιστροφής της readString σε σχέση με τον stack pointer.

3. Αναγκάστε το hello πρόγραμμα να κάνει crash.
Γράψτε ένα πρόγραμμα που ονομάζεται createdata3.c που δημιουργεί ένα αρχείο που ονομάζεται data3, όσο πιο απλό γίνεται, που κάνει το hello program να τερματίσει με segmentation fault (crash). Εξηγήστε στο αρχείο createdata3.c με ένα σύντομο σχόλιο την αρχή λειτουργίας του που προκαλεί το crash του hello.

6. Αναγκάστε το hello πρόγραμμα να τυπώσει "6".
Γράψτε ένα C πρόγραμμα που ονομάζεται createdata6.c που παράγει ένα αρχείο data6, όσο πιο απλό γίνεται, που αναγκάζει το hello πρόγραμμα να τυπώσει το όνομά σας και να προτείνει ως βαθμό το "6". Μπορείτε να δείτε ότι αυτό είναι πολύ εύκολο αν το όνομα σας είναι Angelos Bilas. Ο τρόπος που μπορείτε να αναγκάσετε το πρόγραμμα σε αυτή τη συμπεριφορά είναι να κάνετε overrun τον buffer (buf) της readString και να γράψετε πάνω στη διεύθυνση επιστροφής της readString (στη στοίβα) μια άλλη (κατάλληλη) διεύθυνση μέσα στη συνάρτηση main.

9. Αναγκάστε το hello πρόγραμμα να τυπώσει "9".
Γράψτε ένα C πρόγραμμα που ονομάζεται createdata9.c που παράγει ένα αρχείο data9, όσο πιο απλό γίνεται, που αναγκάζει το hello πρόγραμμα να τυπώσει το όνομά σας και να προτείνει ως βαθμό το "9". Ο τρόπος που μπορείτε να αναγκάσετε το πρόγραμμα σε αυτή τη συμπεριφορά είναι να κάνετε overrun τον buffer (buf) με μια ακολουθία τριών πραγμάτων: (1) το όνομά σας, (2) μια διεύθυνση επιστροφής που δείχνει μέσα στον buffer, και (3) ένα πολύ σύντομο πρόγραμμα σε γλώσσα μηχανής που αποθηκεύει το 9 στη σωστή θέση στη μνήμη και στη συνέχεια κάνει jump στο σωστό σημείο.

Για τα μέρη (6) και (9) αν το όνομά σας είναι πολύ μεγάλο μπορείτε να χρησιμοποιήσετε τους 15 πρώτους χαρακτήρες για την άσκηση.
(extra credit) 10. Αναγκάστε το hello πρόγραμμα να τυπώσει "10". Προσέξτε ότι η printf τυπώνει τον βαθμό ως char. Τι πρέπει να γίνει ώστε να τυπώσει ένα διψήφιο αριθμό?

Παρατηρήσεις για την υλοποίηση:

1. Η άσκηση αυτή θα γίνει σε **Linux/x86** συστήματα του τμήματος (portokali, milo, rodakino, etc.). Για θέματα της αρχιτεκτονικής του x86 και της γλώσσας assembly που χρησιμοποιεί μπορείτε να συμβουλευτείτε την online έκδοση του βιβλίου: Programming from the Ground Up, Jonathan Bartlett 2004. (Τοπικό αντίγραφο του βιβλίου για ευκολία). Ιδιαίτερα χρήσιμα θα βρείτε τα Chapter 3 και Appendix B.
2. Σε κάποια versions του Unix, κάθε φορά που εκτελείται ένα πρόγραμμα χρησιμοποιεί και διαφορετική αρχική τιμή για τον stack pointer. Αυτό κάνει δύσκολο να συντάξει κανείς μια επίθεση όπου η return διεύθυνση βρίσκεται μέσα στη στοίβα. Ωστόσο, το πρόγραμμα hello αντιγράφει το input string από τη μεταβλητή "buf" στη μεταβλητή "Name". Θα δείτε ότι η μεταβλητή "Name" βρίσκεται πάντοτε στην ίδια θέση μνήμης όποτε τρέχει το πρόγραμμα.
3. Σε κάποια versions του Unix, αν προσπαθήσει κανείς να εκτελέσει εντολές από το data section θα προκληθεί segmentation violation. Αυτό συμβαίνει γιατί οι σελίδες μνήμης του data section είναι by default προστατευμένες από το λειτουργικό σύστημα και δεν έχουν execute permission. Το "mprotect" system call στο πρόγραμμα hello, αλλάζει αυτή ακριβώς την συμπεριφορά.
4. Αν προσπαθήσετε πολύ περισσότερο μπορείτε να κάνετε το πρόγραμμα να τυπώσει "10". Η άσκηση δεν σας ζητάει κάτι τέτοιο για να πάρετε το μέγιστο βαθμό.
5. Αν προσπαθήσετε πολύ πολύ περισσότερο, μπορείτε να δημιουργήσετε input το οποίο όταν δοθεί στο πρόγραμμα hello θα λαμβάνει τον έλεγχο της process αυτού που τρέχει το πρόγραμμα σας και να προκαλέσετε κάθε είδους ζημιά. ΜΗΝ ΤΟ ΚΑΝΕΤΕ ΑΥΤΟ! Κάτι τέτοιο αποτελεί παράβαση της υπάρχουσας νομοθεσίας για το ηλεκτρονικό έγκλημα.

Makefile
Συμπληρώστε το Makefile το οποίο υπάρχει στο repo ώστε να περιέχει τουλάχιστον τα εξής targets:

- make clean: σβήνει τα προσωρινά αρχεία, τα executables, και τα dataX input αρχεία.
- make data{3,6,9}: Δημιουργεί το αρχείο data{3,6,9} αντίστοιχα.
- make all: Δημιουργεί όλα τα αρχεία data{3,6,9}.

Logistics
Βήμα 1: Fork repository

- Κάντε fork το repository assignment6 από την ομάδα του μαθήματος στο csd gitlab. Στη συνέχεια αλλάξτε τα permissions σε private όπως αναγράφει στα Policies. Προσθέστε ως members στο repo σας τους TAs του μαθήματος.

Βήμα 2: Source files
Χρησιμοποιήστε τον αγαπημένο σας κειμενογράφο (emacs/vim/nano) για να συμπληρώσετε τα source αρχεία που βρίσκονται κάτω από το φάκελο src (createdata3.c, createdata6.c, createdata9.c, traces).

Βήμα 3: README.md

- Γράψτε το πρόγραμμα σας χρησιμοποιώντας τα εργαλεία gcc, emacs, gdb.
- Χρησιμοποιήστε τον gcc με τις command line παραμέτρους "-Wall, -ansi, -pedantic" για να κάνετε preprocess, compile, assemble, και link το πρόγραμμά σας.
    - Επεξεργαστείτε το README.md text file είναι στο repo ώστε να περιέχει:
    - Το όνομά σας
    - Πράγματα που χειρίζεστε με διαφορετικό τρόπο από ότι ορίζει η άσκηση.

    - Μια περιγραφή της βοήθειας που είχατε από άλλους στη δημιουργία του προγράμματος σας, και σε συμφωνία με το *Policies* section του web page του μαθήματος.
    - (Προαιρετικά) Μία ένδειξη του πόσο χρόνο αφιερώσατε για την άσκηση.
    - (Προαιρετικά) Οτιδήποτε άλλο θέλετε να αναφέρετε
- Σχόλια που περιγράφουν τον κώδικά σας δεν πρέπει να υπάρχουν στο readme file. Πρέπει να τα ενσωματώσετε στο κατάλληλο σημείο του προγράμματος σας.
- Παραδώστε τα αρχεία της άσκησης: createdata{3,6,9}, traces, readme, Makefile.

Βήμα 4: Submit
Η παράδοση της άσκησής σας θα γίνει μέσω git, σύμφωνα με τις οδηγίες που περιγράφονται στο policies. Συγκεκριμένα το repository σας στο gitlab θα πρέπει να είναι fork του repository assignment6 και θα πρέπει να προσθέσετε ως members τους TAs του μαθήματος. Σιγουρευτείτε ότι ο κώδικάς σας έχει γίνει σωστά commit και ότι φαίνονται στο online repository στο account σας στο csd-gitlab. Όταν όλα είναι έτοιμα και έχετε τελειώσει με την άσκησή σας βάλτε το tag assignment6 χρησιμοποιώντας την εντολή:

git add tag assignment6

και κάντε upload το tag χρησιμοποιώντας την εντολή:

git push origin --tags

Προσοχή! Μην κάνετε commit object και executables αρχεία.

Επειδή η εξέταση θα γίνει στα μηχανήματα του Τμήματος θα πρέπει να κάνετε clone το repository στα μηχανήματα του Τμήματος και να κάνετε compile και run τις ασκήσεις σας σε αυτά τα συστήματα. Αυτή είναι η ίδια διαδικασία που θα ακολουθηθεί την ημέρα της εξέτασης.

Στην προθεσμία της παράδοσης ένα script θα τρέξει και θα κατεβάσει όλα τα repositories που έχουν γίνει fork. Αυτά είναι τα repositories που θα βαθμολογηθούν.

Βαθμολογία
Η βαθμολογία θα βασιστεί και στην ορθότητα αλλά και στο σχεδιασμό, όπως αναφέρεται στη σελίδα Policies του μαθήματος. Όπως πάντα, η κατανόηση της άσκησης είναι σημαντικό μέρος του σχεδιασμού.

Last Modified: 12-02-2021 21:12