3η Εργαστηριακή Άσκηση Λειτουργικών Συστημάτων

Γιώργος Ξυπόλιτος 03113629
Στέφανος-Ευστράτιος Πετρίδης 03113418

22-6-2017

Κρυπτογραφική συσκευή VirtIO για QEMU-KVM

Σκοπός της άσκησης

Σκοπός του πρώτου μέρος της άσκησης είναι η υλοποίηση ενός chat το οποίο επικοινωνεί με άλλους clients μέσω TCP/IP. Σε αυτό το μέρος, τα μηνύματα μεταξύ των clients διαδίδονται ως plain text μέσω ενός server. Δηλαδή, αφού γράψει κάποιος client κάποιο μήνυμα, το μεταφέρει στον server και αυτός με την σειρά του το στέλνει σε όλους τους συνδεδεμένους clients.
Στο δεύτερο μέρος της άσκησης προστίθεται η κρυπτογράφηση των μηνυμάτων πριν την αποστολή τους. Η λειτουργία του server παραμένει ως έχει, ενώ πλέον οι clients είναι επιφορτισμένοι με την κρυπτογράφηση και αποκρυπτογράφηση των μηνυμάτων με χρήση ενός predefined key.
Σκοπός του τρίτου μέρους της άσκησης είναι η υλοποίηση ενός paravirtualised driver συσκεύης για την εικονική μηχανή QEMU-KVM. Συγκεκριμένα, ο οδηγός θα αποτελούταν από ένα frontend μέρος, το οποίο εκτελείται στο kernelspace της guest(virtual) machine, και ένα backend μέρος, το οποίο εκτελείται στο userspace της host machine. Ενώ πρακτικά το guest machine τρέχει στο host userspace adress, το adress space του kernelspace του guest machine είναι διαφορετικό από το adress space του userspace του host, κατά ένα offset.

1ο Μέρος

Unencrypted Chat

Σε αυτό το μέρος της άσκησης ζητείται η απλή σύνδεση ενός client-side προγράμματος με έναν server μέσω socket. Από την πλευρά του ο client προσπαθεί μέσω της connect θα συνδεθεί στον server σε μια προκαθορισμένη διεύθυνση και port, ενώ από την πλευρά του ο server ακούει για πιθανές δοκιμές σύνδεσης σε συγκεκριμένο socket μέσω της συνάρτησης listen.
Για την ομαλή λειτουργία του chat απαραίτητη ήταν η χρήση της select. Η select χρησιμοποιείται μαζί με ένα set από file descriptors τα οποία συνεχώς ελέγχει για το αν μπορεί να γράψει ή να διαβάσει σε αυτά. Κάθε φορά που καλείται επιστρέφει τα sets τροποποιημένα, ώστε πλέον να περιέχονται μόνο αυτά στα οποία άμα γίνει κλήση read ή write (ανάλογα σε ποιο set ανήκουν) να μην μπλοκάρουν.
Η υλοποίηση του chat υποστηρίζει πολλούς χρήστες, αν και δεν έγινε υλοποίηση ώστε μαζί με τα μηνύματα των χρηστών να στέλνεται και κάποιο αναγνωριστικό τους(το οποίο θα μπορούσε να ενσωματωθεί μέσω του file descriptor, το οποίο κρατάται για τον κάθε χρήστη).
Για την υποστήριξη πολλαπλών χρηστών, αρκεί η μεταφορά της κλήσης συστήματος accept στο εσωτερικό του main loop του προγράμματος του server, η οποία είναι αυτή που δέχεται καινούριες συνδέσεις από clients στον server. Άμα επιτύχει, τότε προσθέτει τον καινούριο file descriptor στο υπάρχον set, ώστε να διαβάζει και να γράφει και σε αυτόν.

2ο Μέρος

Encrypted Chat

Σε αυτό το μέρος της άσκησης επεκτείνεται το υπάρχον chat, ώστε πριν την μετάδοση των μηνυμάτων να κρυπτογραφούνται και κατά την παραλαβή τους να αποκρυπτογραφούνται. Ορίζουμε ένα κοινό κλειδί για όλους τους clients καθώς και το initialisation vector, μέσω των οποίων εκτελούνται και οι λειτουργίες (άπο)κρυπτογράφησης.
Για την (άπο)κρυπτογράφηση των μηνυμάτων, χρησιμοποιείται ένας έτοιμος driver, ο cryptodev, ο οποίος μέσω εντολών ioctl είτε αξιοποιεί κάποια κρυπτογραφική συσκευή, αν αυτή υπάρχει φυσικά στο hardware του υπολογιστή, είτε προσομοιώνει την λειτουργία του, μέσω αλγορίθμων.
Η λειτουργία του server παραμένει ως έχει, με την διαφορά ότι πλέον είναι απαραίτητη κατά την μετάδοση μηνυμάτων, το μέγεθος των πακέτων να είναι σταθερό και συγκεκριμένου μεγέθους, ώστε να γίνεται σωστή αποκρυπτογράφηση κατά την παραλαβή τους.

3ο Μέρος

VirtIO-Crypto

Frontend

Το μέρος του frontend εκτελείται στο kernelspace της guest machine. Το κομμάτι αυτό παίρνει τα δεδομένα που του στέλνει ο χρήστης όταν καλεί τις αντίστοιχες κλήσεις συστήματος από το userpace της guest machine και τα κάνει map στο adress space του kernel της guest machine.
Υπάρχουν 3 βασικές συναρτήσεις που υλοποιούνται από το frontend:
Η open η οποία είναι επιφορτισμένη με την αρχικοποίηση των fields στο struct που κρατιέται η πληροφορία για την ανοιχτή συσκευή του host και στέλνει στον host κατάλληλα αρχικοποιημένες scatter gather lists, ώστε να γίνει η κλήση συστήματος open στον host. Αφού ¨γυρίσει' τα δεδομένα ο host, ελέγχουμε άμα το file descriptor είναι σωστό και το τοποθετούμε στο αντίστοιχο struct.
Η release η οποία υλοποιεί την αντίστοιχη close του file descriptor που δόθηκε στην συσκευή κατά την κλήση της open και ελευθερώνει την αντίστοιχη μνήμη που δεσμεύθηκε κατά την δημιουργία του struct της ανοιχτής συσκευής.
Η ioctl η οποία είναι επιφορτισμένη με πολλαπλές λειτουργίες, ανάλογα με το ioctl που εκτελεί ο χρήστης. Η συγκεκριμένη λειτουργία που θα εκτελεστεί εξαρτάται από το δεύτερο όρισμά της το unsigned int cmd. Οι λειτουργίες της είναι:

- CIOCGSESSION: Η λειτουργία αυτή ξεκινάει ένα session με την συσκευή και απαιτεί από τον χρήστη την είσοδο του κλειδιού που θα χρησιμοποιηθεί ως κλειδί για τις (από)κρυπτογραφήσεις, τον αλγόριθμο που θα χρησιμοποιηθεί και άλλες πληροφορίες που περιέχονται στο struct session_op.

- CIOCFSESSION: Η λειτουργία αυτή κλείνει ένα ανοιχτό session με την συσκευή το οποίο γίνεται identified από τον αριθμό που του δόθηκε κατά το άνοιγμα του με την κλήση της CIOCGSESSION.
- CIOCCRYPT: Η λειτουργία αυτή κρυπτογραφεί ή αποκρυπτογραφεί τα δεδομένα που δίνονται στο πεδίο src και τα τοποθετεί έπειτα στο dst. Η λειτουργία που θα πρέπει να επιτελέσει περιέχεται στο struct crypt_op στο πεδίο op.

Σε όλες τις παραπάνω κλήσεις απαιτούνται κατάλληλες χρήσεις των συναρτήσεων copy_from_user και copy_to_user ώστε να γίνεται σωστό mapping των δεδομένων καθώς μεταφερόμαστε στα διάφορα adress spaces.

Backend

Το μέρος του backend εκτελείται στο userspace της host machine και είναι αυτό το οποίο καλεί τις πραγματικές κλήσεις συστήματος του οδηγού. Μέσω της χρήσης της virtqueue και scatter gather λιστών, το backend παίρνει δεδομένα από το frontend, τα χρησιμοποιεί για να εκτελέσει τις κατάλληλες κλήσεις συστήματος στην host machine και επιστρέφει πάλι μέσω virtqueue τα δεδομένα στο frontend.
Για την αναγνώριση της κλήσης που γίνεται, έχουμε αρχικοποιήσει κατάλληλα την syscall_type scatter gather list.
Για την κλήση των πραγματικών ioctl απαιτείται η χρήση struct που βρίσκονται στο stack του backend. Επίσης οποιοσδήποτε pointer σε δομή πρέπει να περαστεί ξεχωριστά σε διαφορετική scatter gather list ώστε να γίνει το σωστό mapping του.
Το offset που απαιτείται για το mapping των δεδομένων από το kernelspace της guest machine στο userspace της host machine τοποθετείται από την συνάρτηση virtqueue_get_buf.
Σημαντική είναι η παρατήρηση ότι εφόσον η virtqueue_get_buf εκτελείται με busy wait, η εκτέλεση της virtqueue_notify είναι περιττή, κάτι το οποίο γίνεται αισθητό και από το γεγονός ότι η συνάρτηση vq_has_data του αρχείου crypto-module.c δεν κάνει τίποτα.