Francesco Soncina

DOCUMENTO FINALE PROGETTO VSD

2014

Indice

1 UMsandnet **5**
1.1 Descrizione 5
1.2 Funzionamento 5
1.3 Possibili utilizzi 6
1.4 Sviluppi futuri 6

Capitolo 1

UMsandnet

1.1 Descrizione

UMview permette di creare delle macchine virtuali a livello di syscalls, ovvero l'esecuzione dentro alla macchina è identica all'esecuzione fuori, tranne per le chiamate a sistema, che vengono intercettate dal core di *UMview* e delegate ad eventuali moduli e sottomoduli per gestirne la virtualizzazione. *UMsandnet* è un modulo per *UMview* che permette di supervisionare in modo interattivo la connettività di un singolo processo, ovvero la possibilità di decidere se consentire traffico in uscita od entrata. Il funzionamento del modulo si basa sull'intercettazione di determinate socketcalls, lasciando la decisione della loro effettiva esecuzione all'utente, permettendo quindi di avere una sorta di firewall prototipale in userspace per singoli processi.

1.2 Funzionamento

Per poter utilizzare il modulo è sufficiente compilarlo e copiarlo nella directory dei moduli di *UMview* e caricarlo attraverso l'utility *um_add_service*, la quale richiede il nome del modulo come argomento (e.g. *um_add_service umsandnet*). La compilazione e l'installazione possono essere eseguiti attraverso il Makefile fornito. Le socketcalls principali interessate sono le seguenti:

- *int socket(int domain, int type, int protocol)*: è possibile approvare/negare traffico di basso livello, i.e. domain == AF_PACKET || type == SOCK_RAW, che altrimenti sfuggerebbe alle socketcalls seguenti.
- *int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen)*: è possibile approvare/negare tutte le connect() oppure solo per certi ip.

- *int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen)*: è possibile approvare/negare tutte o le singole bind().

1.3 Possibili utilizzi

Questo modulo è stato pensato per dotare l'utente di un controllo basilare sulla connettività di un singolo processo sfruttando la virtualizzazione parziale offerta da *UMview*, ovvero intercettando le socketcalls prima che vengano effettivamente gestite dal kernel, permettendole o negandole a seconda dei casi. Per quanto riguarda la rete, il processo eseguito dentro la macchina virtuale parziale non ha un accesso diretto con il kernel e quindi si possono avere tutti i vantaggi di aver installato un firewall nel sistema, ma eseguendo solo codice utente e solo per quella specifica macchina virtuale. Se si vuole eseguire un processo senza che abbia la possibilità di usufruire della rete, dovremmo, se per esempio non si ha a disposizione un firewall, staccare la rete a tutta la macchina virtuale, per essere sicuri che quel processo non riesca in nessun modo ad utilizzare la rete, poiché esso ha un contatto diretto col kernel come tutti gli altri processi attivi; al contrario la virtualizzazione parziale ci fornisce un metodo comodo (solo codice userspace) ed efficiente (viene virtualizzato solo l'accesso alla rete) per separare la connettività di un processo (la macchina virtuale) da quella del sistema host. Si potrebbe anche essere interessati a consentire tutto il traffico di un processo tranne verso certi ip, senza dover negare il traffico verso quell'ip a tutto il sistema attraverso *iptables*.

1.4 Sviluppi futuri

UMsandnet si dedica solo al controllo della rete, quindi in futuro potrebbe trovare un ruolo complementare ad altri moduli scritti con lo stesso obbiettivo che però potrebbero controllare la creazione, la modifica e la rimozione di files oppure l'esecuzione di altre syscalls critiche, in modo da poter trasformare UMview in una sorta di sandbox interattiva dove l'utente possa decidere, prima che lo faccia il kernel, se un processo debba, o no, eseguire certe syscalls che in date situazioni potrebbero essere ritenute critiche.