Università di Pisa

DIPARTIMENTO DI INFORMATICA

GESTIONE RETI

Anno Accademico: *2018/2019* ***DOCENTE:*** *Luca Deri*

AUTORE
Alessio Russo
Stefano Torneo

RELAZIONE

Version 1.4

Contenuti

1. Descrizione Progetto

Lo scopo del progetto consiste nel calcolo della fingerprint nel formato MD5 per identificare e verificare l'identità dei Client e Server durante una comunicazione SSH. Se la fingerprint cambia, la macchina potrebbe aver cambiato la chiave pubblica, per esempio installato una nuova applicazione, ma potrebbe soprattutto indicare che la macchina a cui ti stai connettendo allo stesso indirizzo è cambiata. Per esempio, a causa di un attacco man-in-the-middle, dove l'attaccante intercetta o ridireziona il traffico ssh su un host diverso che potrebbe scoprire usr/pwd.

2. Scelte di Progettazione

Il progetto è stato realizzato in C su Linux versione Ubuntu 18.04 .
La libreria principale usata è la libpcap per la cattura dei pacchetti.

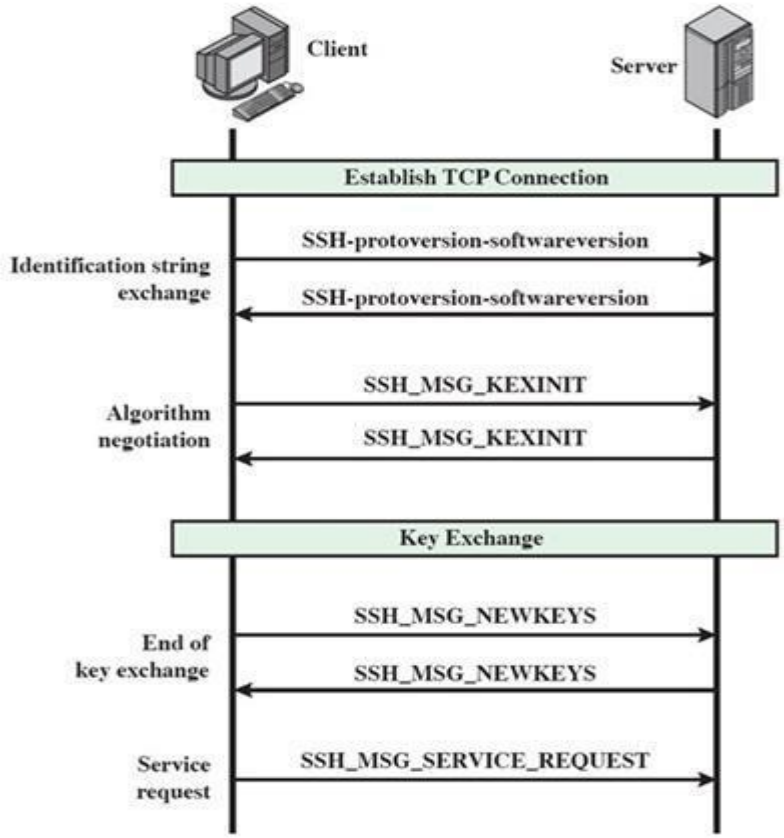
2.1.1. Strutture dati usate

La struttura dati principale usata è un array dedicato alla raccolta di tutte le informazioni che vengono estratte dai pacchetti. Un elemento dell'array rappresenta una comunicazione ssh tra client e server.
Un elemento è un record di tipo "SSH" da noi definito.
Una comunicazione SSH è suddivisa in 2 parti: informazioni che riguardano il client e quelle che riguardano il server.
Per completare un elemento bisogna ricevere 4 tipi di pacchetto(vedi 2.1.2).
Ogni comunicazione SSH è identificata dalla quadrupla:

<ip sorgente, porta sorgente, ip destinazione, porta destinazione>.

Quando arriva un nuovo pacchetto, nel caso non venga scartato, si estrae la quadrupla e si effettua una scansione lineare per capire se si tratta di un pacchetto appartenente ad una nuova comunicazione o di una già iniziata e si aggiornano le informazioni.

2.1.2. Struttura Pacchetti SSH



In figura è mostrato lo scambio dei pacchetti interessati

Ci sono 2 tipi di pacchetti che devono essere catturati:

1. SSH-protoversion-softwareversion
2. SSH_MSG_KEXINIT

Il primo contiene informazioni che permettono di identificare la versione del protocollo e del software, uno per il client e uno per il server.

```
2664 67917.002427  192.12.193.11   192.12.194.254   SSHv2    82 Client: New Keys
2654 67916.966641  192.12.193.11   192.12.194.254   SSHv2   109 Client: Protocol (SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.6)
 171 4043.949138   192.12.193.27   192.168.17.17    SSHv2    87 Client: Protocol (SSH-2.0-OpenSSH_6.9)
1025 9775.334397   192.12.193.27   192.168.17.17    SSHv2    87 Client: Protocol (SSH-2.0-OpenSSH_6.9)
  64 1960.561778   192.12.192.93   192.12.193.3     SSHv2    89 Client: Protocol (SSH-2.0-check_ssh_1.5)
 126 3160.495710   192.12.192.93   192.12.193.3     SSHv2    89 Client: Protocol (SSH-2.0-check_ssh_1.5)
```

Esempio figura: SSH-2.0-OpenSSH_6.9

Con il secondo tipo di pacchetto, sia client che server si scambiano la lista dei nomi degli algoritmi supportati (vedi RFC 4253 per maggiori dettagli).

2658	67916.980972	192.12.193.11	192.12.194.254	SSHv2	2034 Client: Key Exchange Init
2660	67916.981308	192.12.194.254	192.12.193.11	SSHv2	1698 Server: Key Exchange Init

Per individuare questo pacchetto basta controllare il valore del campo Message code che deve essere uguale a 20. Per ottenere gli algoritmi basta prendere le rispettive lunghezze e spostarsi di conseguenza sul payload del pacchetto.

```
▼ SSH Protocol
  ▼ SSH Version 2 (encryption:aes128-ctr mac:hmac-md5-etm@openssh.com compression:
      Packet Length: 1964
      Padding Length: 8
    ▼ Key Exchange
        Message Code: Key Exchange Init (20)
      ▼ Algorithms
          Cookie: 951408fc8d0ac9709e8c8a0859bc619d
          kex_algorithms length: 212
          kex_algorithms string: curve25519-sha256@libssh.org,ecdh-sha2-nistp25
          server_host_key_algorithms length: 359
          server_host_key_algorithms string [truncated]: ecdsa-sha2-nistp256-ce
          encryption_algorithms_client_to_server length: 233
          encryption_algorithms_client_to_server string [truncated]: aes128-ctr
          encryption_algorithms_server_to_client length: 233
          encryption_algorithms_server_to_client string [truncated]: aes128-ctr
          mac_algorithms_client_to_server length: 402
          mac_algorithms_client_to_server string [truncated]: hmac-md5-etm@open
          mac_algorithms_server_to_client length: 402
          mac_algorithms_server_to_client string [truncated]: hmac-md5-etm@open
          compression_algorithms_client_to_server length: 26
          compression_algorithms_client_to_server string: none,zlib@openssh.com
          compression_algorithms_server_to_client length: 26
```

Struttura del payload del pacchetto che contiene gli algoritmi. Le informazioni necessarie da estrapolare sono:

- *Packet Lenght*
- *Padding*
- *Message code*
- *Kex_algorithm lenght, server_host length, ecc.*

3. Struttura logica del codice

Sorgenti presenti:

- **Filter.c** : sorgente principale
- **MD5ndpi.c** : funzioni per MD5 hash di ntop
- **MakeFile** : file per compilazione automatica
- ***.pcap** : file pcap di prova contenenti pacchetti ssh

Il codice è strutturato con una serie di funzioni che servono per catturare le informazioni necessarie da visualizzare a video.
Inizialmente nella funzione Main vengono inizializzate le variabili e controllati i parametri passati dall'utente.
In base ai parametri passati, se sono validi, si passa alla modalità online o offline.
Viene richiamata la funzione my_packet_handler a cui viene passato il puntatore al pacchetto, quest'ultimo ci permette di accedere ai campi per ottenere le informazioni.
Qui vengono prelevate inizialmente le lunghezze dell'header Ethernet, IP e TCP, necessarie al calcolo della lunghezza del payload.
Dopodiché, in base alla lunghezza del pacchetto e ai primi caratteri del payload, si individua il tipo di pacchetto.
Nel caso in cui il pacchetto è di tipo SSH-protoversion-softwareversion allora si richiama la funzione GetPos per ottenere la posizione in cui devono essere inserite le informazioni appena ricevute, poi viene richiamata la GetSSHProtocol per ottenere il payload che rappresenta protocollo e versione utilizzata in quella comunicazione. Vengono inoltre memorizzati IP e porta di sorgente e destinazione.

Nel caso in cui il pacchetto è di tipo SSH_MSG_KEXINIT, individuato in base alla lunghezza e al valore del campo Message Code, allora si richiama sempre la GetPos, poi la funzione Algorithms per ottenere l'insieme degli algoritmi ssh utilizzati nella comunicazione e infine avviene il calcolo della fingerprint mediante l'utilizzo delle funzioni messe a disposizione da NTOP.

4. Calcolo Fingerprint

Il calcolo della fingerprint consiste inizialmente nella cattura degli algoritmi utilizzati.
Gli algoritmi presi si riferiscono ai seguenti campi del pacchetto:

- kex_algorithms string
- encryption_algorithms_client_to_server
- mac_algorithms_client_to_server
- compression_algorithms_client_to_server

Dopodiché gli algoritmi vengono concatenati separandoli da ';' e poi utilizzando le funzioni MD5 opportune si ottiene la fingerprint.

Campi	Algoritmi presi
Key Exchange methods	diffie-hellman-group-exchange-sha256,diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1,diffie-hellman-group1-sha1
Encryption	aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,aes192-cbc,aes256-cbc,arcfour,rijndael-cbc@lysator.liu.se
Message Authentication	hmac-md5,hmac-sha1,umac-64@openssh.com,hmac-ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-md5-96
Compression	none,zlib@openssh.com

Fase raccolta algoritmi

Concatenazione	Fingerprint
diffie-hellman-group-exchange-sha256,diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1,diffie-hellman-group1-sha1;aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,aes192-cbc,aes256-cbc,arcfour,rijndael-cbc@lysator.liu.se;hmac-md5,hmac-sha1,umac-64@openssh.com,hmac-ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-md5-96;none,zlib@openssh.com	**c1c596caaeb93c566b8ecf3cae9b5a9e**

Fase di concatenazione e calcolo fingerprint

5. Gestione segnali

Quando si esegue il codice in modalità live, indicando l'interfaccia da usare, il programma analizzerà ogni pacchetto che riceve. Le eventuali informazioni però possono essere visualizzate solo alla terminazione del processo.
Per interrompere il processo in sicurezza si possono inviare i seguenti segnali:

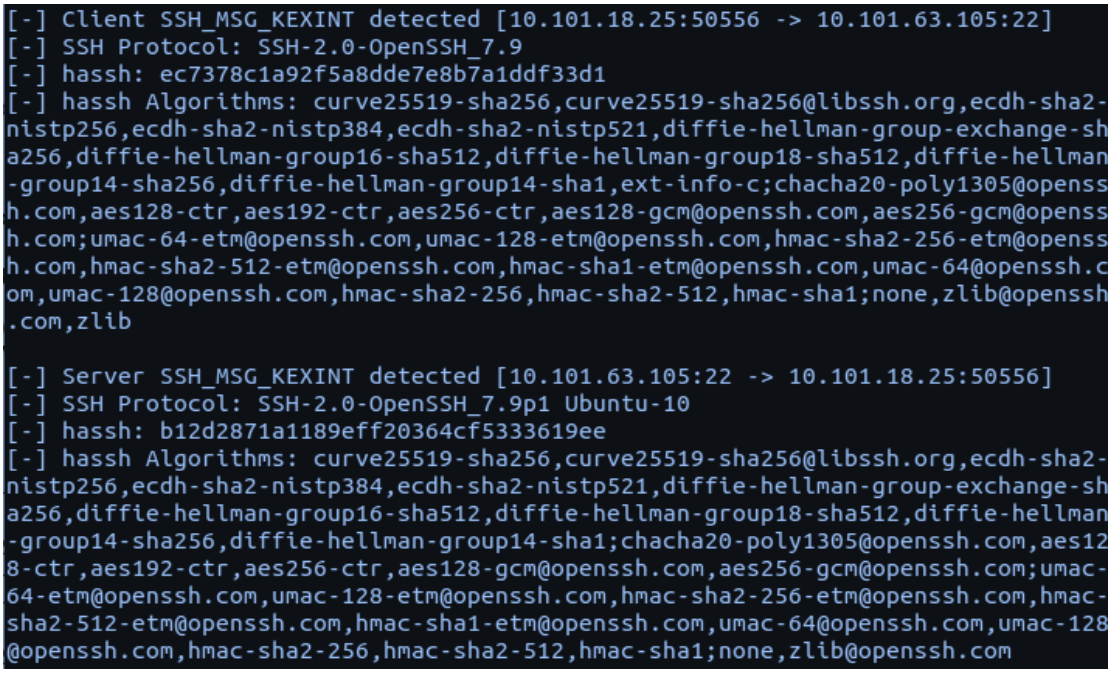
- SIGINT
- SIGQUIT
- SIGSTOP

Per i segnali elencati è stato installato un signal handler che termina il loop in sicurezza e il processo terminerà in maniera safe, visualizzando i risultati ottenuti.

6. Manuale

Per la compilazione basta eseguire il Makefile digitando da shell ‘make’.
Se si vuole eseguire un test con un pcap, quindi in modalità offline, basta digitare ‘make test1’, altrimenti per eseguire il programma è necessario digitare ‘./filter’ indicando un’interfaccia (modalità online) o un pcap (modalità offline).
È possibile eseguire il programma in due modalità differenti in base ai parametri passati al momento dell’esecuzione:

- in modalità online vengono catturati i pacchetti che arrivano durante l’esecuzione, quando si desidera visualizzare le informazioni allora basta premere la combinazione dei tasti ‘Ctrl+C’ per interrompere la cattura e passare alla stampa (es. ‘./filter lo’, per catturare i pacchetti sull’interfaccia loopback)
- in modalità offline invece vengono catturati i pacchetti salvati nel pcap passato come parametro quindi la stampa delle informazioni raccolte è immediata (es. ./filter ssh.pcap, per catturare i pacchetti dal file .pcap indicato).



Un esempio di stampa delle informazioni ottenute