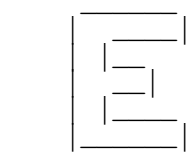
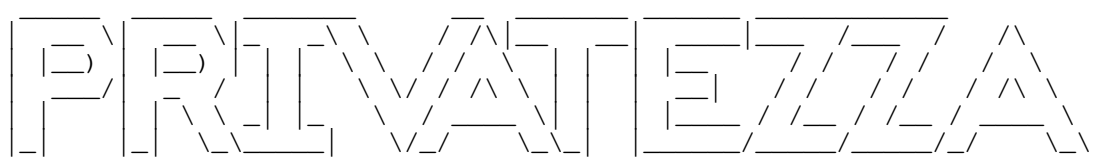

==========

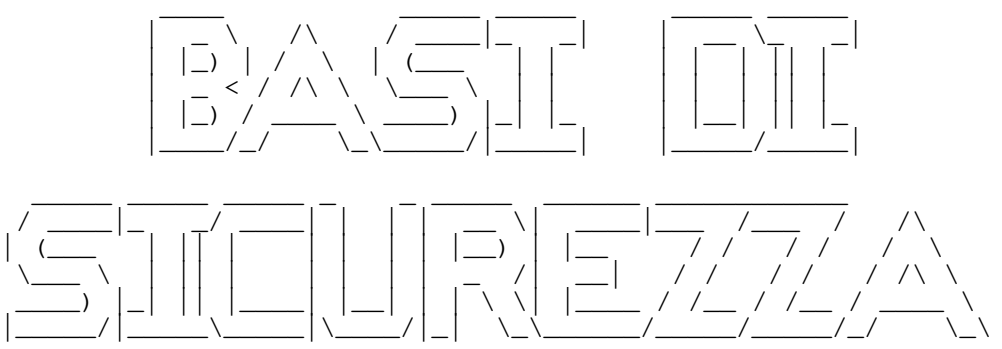




CORSO 2023

Giacomo Castellana

==========



FONT ASCII TITLES BIG
da copiare regole auree

POLITICHE DI ACCESSO

Discretionary Acces Control (DAC)
- policy dettata dal proprietario della risorsa che decide come gestire gli accessi (uso classico del pc)

Mandatory acces control (MAC)
- policy dettata dal sistema e decisa per tutti a priori (ambiente militare)

Role Based Acces Control (RBAC)
- a seconda della posizione e del ruolo la policy cambia (studenti, prof, sysadmin)

DEFINIRE UNA POLITICA DI SICUREZZA

Matrice che associa un utente ai permessi per ogni risorsa

s\o	car	tas	...
us1	x	-	...
us2	-	-	...
us3	x	x	...
...	...	...	...

Per tanti utenti non è gestibile --> COME POSSIAMO SFRUTTARE LE NOSTRE CONOSCENZE PER COMPRIMERE LA STRUTTURA DATI SENZA PERDERE PEZZI?

* divido le persone/utenti secondo il loro ruolo e posizione (se ho 50k utenti ma essi fanno parte di 5 gruppi da 10k -> anzichè avere 50k linee ne avrò 5)

ACCES CONTROL LIST ACL

g\o	car	tas	...
gr1	x	x	...
gr2	-	-	...
...	...	...	...

DA TENERE IN UN LUOGO SICURO PER TENERE SANA E SALVA LA POLICY -
in Linux/Unix è contenuta nella I-NODE
I-NODE -> struttura dati su HD in struttura chiamata I-NODE TABLE e contienete i metadati dei file

CAPABILITY

garantisce l'accesso attraverso dei TOKEN
tenuta nel KERNEL dell'OS per mantenere la protezione alta -> se qualcuno ne entra in possesso E' quella persona per il computer -> IMPORANTE
usato in Windows

Una banca tiene traccia di chi ha l'accesso alla cassetta di sicurezza, deve tenere al sicuro l'elenco -> ACCES CONTROL LIST

AUTENTICAZIONE - la banca controlla chi entra
PARTECIPAZIONE - la banca partecipa
FALSIFICAZIONE - l'elenco è al sicuro
DELEGAZIONE - bisogna aggiunger euna entry ma è complesso

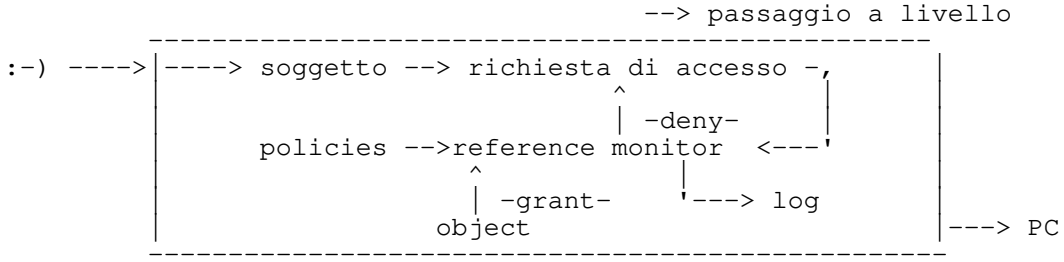
ELIMINAZIONE - è facile da rimuovere una entry

Io ho le chiavi della cassetta e posso decidere a chi darle, se voglio darle -> CAPABILITY

AUTENTICAZIONE - la banca non controlla
PARTECIPAZIONE - la banca non partecipa
FALSIFICAZIONE - la chiave non è modificabile/copiabile
DELEGAZIONE - la chiave può essere data a chiunque
ELIMINAZIONE - è difficile farsi dare la chiave indietro

ACL E CAPABILITY HANNO PRO E CONTRO INVERSI
--> ACL USATA DI PIU'

Il controllo di accesso è gestito dal Reference Monitor, un software che gesticisce OGNI accesso e il cui funzionamento è ASSICURATO al 100%



Uno se non il PRIMO SO

L'UTENTE IN UNIX

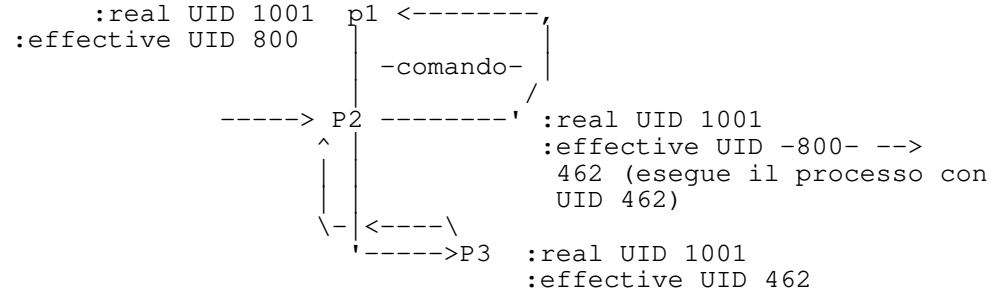
Tutti coloro che sono presenti nel file /etc/passwd
username:password:UID:GID:name:homedir:shell
UID - User ID - numero 16b (per root è 0)
GID - Group ID - numero 16b (per root è 0)
password contiene x ora perché le password sono in /etc/shadow

root ha controllo totale tolte:
* LA CONOSCENZA delle password, ma può RISCRIVERLE
* la RISCRITTURA dei file Read Only

/etc/group conteiene groupname:password:GID:list_of_users

SOGGETTI

Processi distinti da PID -> generati da exec e fork -> oguno di loro ha un UID/GID reale e un UID/GID effettivo
effective UID è usato per gestire i processi



Ad ogni oggetto possono accedere 3 tipi di utente
* OWNER
* GROUP
* OTHER - qualunque altra cosa

Ogni soggetto può avere 3 operazioni --> si necessita di 3 bit
* READ r
* WRITE w
* EXECUTE x

+ 7 = 111 = TUTTO
+ 6 = 110 = solo read and write
+ 5 = 101 = solo read e execute
+ 4 = 100 = solo read
+ 3 = 011 = solo write e execute
+ 2 = 010 = solo write
+ 1 = 001 = solo execute

chmod fa modificare i diritti di accesso
sia con numeri sia con parole
sudo chmod 0754 file.txt ->user tutto, group leggi esegui, other solo leggi

Ogni utente ha una sua directory di cui possiede tutti i permessi (rwx) e può modificare i permessi degli altri utenti per accedervi

GETFACL mostra nome,proprietario,gruppo e file ACL esistente
getfacl nome_file_o_cartella

SETFACL modifica ACL
setfacl -m u:nome_utente:xxx nome_file_o_cartella - xxx può essere numero (7,6,5,..,1) o rwx,rw-,r--,...,--x
setfacl -m g:nome_gruppo:xxx nome_file_o_cartella
setfacl -m o

setfacl -x u:..... - rimuove i permessi

GREP

Permette di cercare stringhe all'interno di file
grep stringa file
Ha opzioni principali
- c conta le righe dove c'è match
- i fa cercare la stringa indipendentemente dal case (hello, HELLO, HellO, heLLo,heLlo,...)
- v stampa le linee
- n stampa il numro di riga

Grep usa anche le espressioni regolari per fare le richerche
[:::] dove ::: sono caratteri -> grep cercherà tutte li stringhe che contengono quei caratteri
'-> indica tutte tranne ciò che c'è dopo

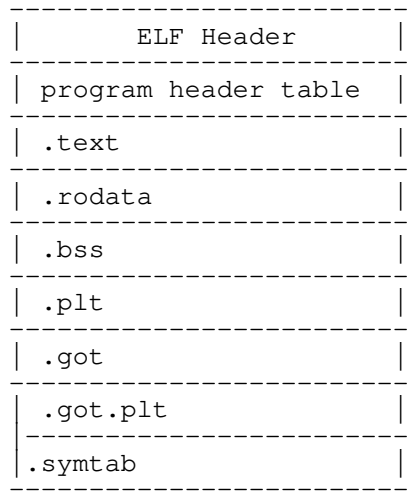
FORMATI DI UN PROGRAMMA

Sorgente -> dato in pasto al compilatore ---> UNA VOLTA dava il codice OGGETTO --> poi lo dava al LINKER---,
|
|--->ESEGUIBILE
|
|
'----> ora fa sia compilazione sia linking assieme

foo.c ---> foo.o ---> foo.out <-- eseguito con un LOADER
comp link

ELF

Executable and Linkable Format ---> formato per eseguibili LINUX, videogiochi Sony, ...
Formato BINARIO
Contiene il codice e TUTTE le info necessarie alla esecuzione
Composto da:
* ELF Header
* Sezioni
* Segmenti



HEADER

readelf -h nome --> permette di leggere il file elf di un eseguibile per leggere le informazioni

e_entry: entry point in memoria centrale
e_phoff: offset della Program Header Table
e_shoff:
e_flags:
.....

SEZIONI
contiene tutte le info per il linking

readelf -S nome

sh_type:
sh_flags:
sh_addr:
sh_offset:
....

SEGMENTI
dividono il file elf in spezzoni per .....

readelf -l nome

p_typr:
p_flags:
....

SORGENTE --> vi,vim,neovim,emacs
ELF FILE --> readelf
OBJECT --> objdump

EXEC

legge il path e cerca il file
lo carica in memoria
verifica se è un elf --> legge l'ELF Header --> legge il MAGIC NUMBER (e altro)
e se è 7f 45 4c 46
se non lo è
* se inizia con #! allor chiama l'interprete corretto
* se è simile a /proc/sys/fs/binfmt_misc allora il kernel chiama il comando di interprete specifico
se lo è:
* se è DYNAMIC-LINKED il kernel legge l'interprete specificato nell'ELF, lo chiama (attraverso il LINKER DINAMICO) per il file e gli da' controllo
* se è STATIC-LINKED il kernel lo carica

readelf -a cookie | grep interpreter fa vedere che interprete è usato
viene allocata la memoria virtuale per
* il binario
* le librerie
* lo heap
* lo stack
* la memoria mappata specificatamente dal programma
* il codice kernel nella prima metà della memoria

Dopo la compilazione il programma non entra subito in esecuzione ma fa dei passi precedenti e poi chiama il main del programma

ARGOMENTI

IN C
argc -> #argomenti
argv -> array contenente gli argomenti
envp -> variabili di ambiente

SYSTEM CALL

Durante l'esecuzione il programma esegue delle Syscall per fare entrare l'os in

modalità KERNEL e fargli eseguire ciò che è necessario.
Per vedere quali system call sono usate in un comando basta usare
strace comando -l
Su Linux ce ne sono 300 -> visibili usando "man 2 open"
Durante l'esecuzione l'SO comunica con i processi usando le SIGNAL
syscall speciali che stoppano il processo
hanno un numero assiociato
9 --> chiamata SIGKILL -->il processo invoca ABORT su se stesso
(fa SO se il processo non si ammazza)

Un processo muore in due modi:
1: fa la exit();
2: riceve SIGKILL

ALL PROCESSES MUST BE R E A P E D

Un processo dienta Zombie quando termina e deve essere rimosso dal padre con wait();
Quando viene rimosso si libera spazio per un altro processo
Se il padre muore prima i processi vengono rimossi da un DAEMON che cambia il loro PID a 1. Il processo padre con PID 1 fa periodicamente la wait();

PRIVILEDGE ESCALATION

COMPLETE MEDIATION
Ogni tanto un processo necessita di più diritti di accesso di quelli che possiede --> Fa una scalata di privilegio, ma deve essere fatta in modo tale che non si creino problemi
In UNIX si chiama Set-UID --> permette a utenti SENZA root che esegue un programma di usufruire dei diritti di accesso del programma quando lo usa
ESEMPIO passwd

```
$ ls -l /usr/bin/passwd
-rwSr-xr-x 1 root root 41284 Sep 12 2012 /usr/bin/passwd
```
passwd è di ROOT, ma quando lo eseguo DIVENTO io ROOT (finchè lo uso)

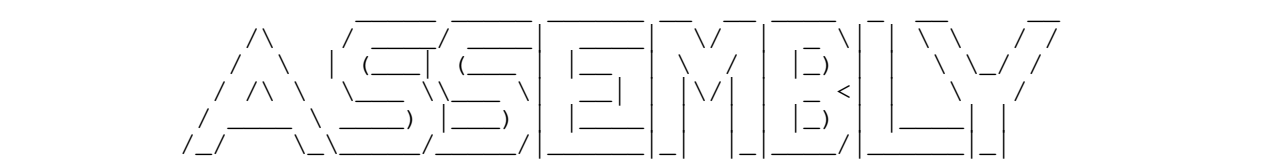
I DUE USER ID

Ogni processo ha DUE UID:
* Real UID --> UID del vero padrone del processo --> quelli su passwd
* Effective UID --> identifica i privilegi di un processo --> l'accesso si basa su di esso

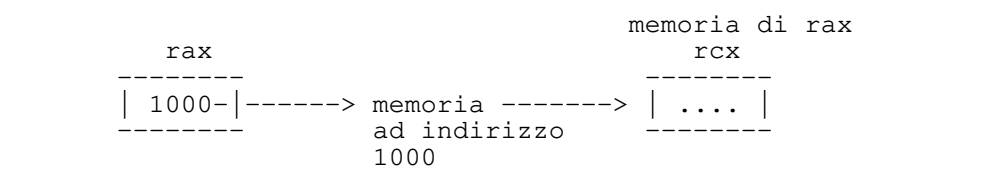
Di norma sono uguali. Quando Set-UID viene eseguito l'EUID cambia a quello di root (0) --> lo si può capire se nei permessi c'é una S (vedi sopra)

BASH

Scripting IN Console
Molto potente



```
LABEL OPCODE OPERAND COMMENT --> unico obbligatorio è opcode
commento ; o # --> noi #
usiamo MASM

dati salvabili in registri o variabili <---- in memoria
                  ^
                  |
               speciali in chip
               UNICI MANIPOLABILI


byte B                  =      ~1/2B~   =       4b
word                    =       1B      =       8b
double word             =       2B      =       16b
quadruple word          =       4B      =       32b
double quadruple word   =       8B      =       64b

registri sia a 64b, che 32b, che 16b che 8b a seconda di quali bit
voglio
        ESEMPIO
               rax = 64b
               eax = 32b
               ax  = 16b -> ah+al (higher e lower)
               al  = 8b

LITTLE ENDIAN!!!!

SOLO DUE TIPI DI DATO
        * NUMERI --> notazione binaria
               + 100 -> 0110 0100
        * CARATTERI --> UTF8 --> OGNUNO OCCUPA 1B
               + 100 -> 00110001 00110000 00110000
                           1        0         0

I DATI HANNO BISONO DI RISERVAMENTO DI SPAZIO IN MEMOORIA
        ESEMPIO
               buffer:         resb    64 #byte
               wordvar:        resw    1  #word
               realarray:      resq    10 #array

                               db      0x55 ->85 numero O 'U'
                               db      'hello'

Per spostare da memoria a registri bisogna usare delle ADDRESSING MODES
        1 Accedere ai dati dai registri
        2 "        "   "   da immediato
        3 "        "   "   da memoria
               a Modalità Diretta
               b "        Indiretta
                       * Registrer addressing -> muove i dati da e in
                         i registri --> mov rdx,rcx
                               + a registro
                                       mov rdx,rcx
                               + a immediato
                                       mov rax,100 --> NON COSI',
                                       è vago, bisogna specificare
                                       mov DWORD PTR rax,100 --> usa
                                       solo i primi 32b
                       * Indirect Addressing --> voglio usare i
                         puntatori --> [registro]
                                       mov rcx,[rax] --> metto in rcx
                                       il contenuto dell'indirizzo di
```



INDIRIZZAMENTO DIRETTO è intuitivo ma poco flessibile
" INDIRETTO non è intuitivo ma è flessibile

LITTLE ENDIAN

Dati salvati al contrario

```
        mov     eax,0xc001ca75  #carico diretto ----> | c0 | 01 | ca | 75 |
        mov     rcx,0x10000     #65536
        mov     [rcx],eax       #carico eax all'INDIRIZZO 65536 in LITTLE ENDIAN
                                --> | 75 | ca | 01 | c0 |
        mov     bh,[rcx]        #carico 0x75 che si trova a 65536 IN MEMORIA
```

GIOCARE CON L'ASSEMBLY

```
ESEMPIO:
        LOOP    add     rax,3
                jmp     LOOP

somma di 20 elementi di array
                xor     rax   #contatore
                xor     rbx   #somma
                lea     rcx,a #inzirizzo array
        sumloop:
                mov
                add
                ....

LEA Load Effective Address -> carica in 1 l'ndirizzo della variabile
        lea     rcx,buffer |       carico in rax il contenuto di buffer
                           |------>INDIRETTAMENTE
        mov     rax,[rcx]  |


add dest,src    #a=a+b
sub dest,src    #a=a-b
inc dest        #++
dec dest        #--
neg dest        #...
cmp dest,src    #confronta (b-a) e poi <0,>0,=0 modifica STATUS REGISTER a
                 seconda

mul     dest,src        #a=a*b SENZA segno
        può essere uno solo, l'altro è supposto in rax
imul    dest,src        #a=a*b CON segno
div
idiv

CONTROLLO DI FLUSSO
opcode label
        controlla tra gli STATUS REGISTER della la cmp
je      #solo se destinazione=origine
jg      #solo se destinazione>origine
jge     #solo se destinazione>=origine
jl      #solo se destinazione<origine
jle     #solo se destinazione<=origine
jmp     #non condizionata

SYSCALL
usabili salvando sul registro rax il numero della syscall, negli altri registri
inserisco gli argomenti e poi chiamo syscall
i ritorni sono eax
        * sys_read      0       rdi=file_descriptor(0 stIn)
                                rsi=caratteri letti
                                rdx=numero caratteri da leggere
        * sys_write     1       rdi=file_descriptor(1 schermo)
```

```
                                rsi=caratteri stampare
                                rdx=numero caratteri da stampare

        leggere 100B da stdin a SP
        mov rdi,0
        mov rsi,rsp #in rsi valore dello SP --> se ho buffer uso lea rsi,buffer
        mov rdx,100
        mov rax,0
        syscall

                                        *SHELL CODE*

Codice in linguaggio macchina --> utile per sfruttare le vulnerabilità usando
                                  le INJECTION
                                         |
                                         v
                                  non sulla Harvard
durante l'esecuzione di un processo si inserisce del codice in più
                                 ^
                                 |
Funziona perché nelle architetture di V.N. la memoria contiene SIA istruzioni
SIA dati --> si possono aggiungere cose in mezzo

        PROGRAMMA DA COPIARE myFirstInjection.c
                        --> può fare segmentation fault (causa sezione dell'IF)
                            uso gdb -q nome
                            la runno finchè non segmenta
                            faccio la info proc map
                            x/s $rip --> 0x7fffffffdb20:        "ciao"
    CONTENUTO della stringa COLPEVOLE
                            x/i $rip --> 0x7fffffffdb20:     movsxd 0x61(%rcx),
                                                             %ebp
                                                             ISTRUZIONE
                                                             COLPEVOLE

        objdump -M intel -d nomeEseg
                vedo la objdump

        objcopy --dump-section .text=RawFile nomeEseg
                copiare l'elf fuori dall'elf

        hexdump -C RawFile
                vedere la hexdump a byte

        shellCodeInjection --> fa aprire una shell a un processo aperto -->
                               con diritti di ROOT
        shellCodeInjection.s --> shellCode
                poi uso objcopy per metterlo su un file con SOLO la parte di
                testo

        HO BISOGNO DI UN PAZIENTE 0 PER TESTARE PRIMA DI INIETTARE --> *CARRIER*
        carrier.c --> carrier --> tiene un pezzo di memoria di 1000B
                                  legge e mette su questo pezzo RawFile
                                  lo esegue

        gcc -o carrier carrier.c

        cat RawFile - | ./carrier

        ESERCIZIOOOOOOO
                fare una iniezione per aprire il file Flag in root usando
                myFirstInjection --> done and dusted


                                *MEMORY ERROR EXPLOITS*

S M A S H I N G  the stack

Usata da Morris per l'Internet Worm

-citazione di Aleph One-
E' possibile corrompere lo stack con l'overflow di un array e causare un salto
ad una routine casuale
```

```
        ---------------------
        |    kernel space   |
        ---------------------
        |     stack         | --> roba in main --> int a,b;
        ---------------------
        |     heap          | --> roba in heap (malloc,...) -->
        |                   |     int *ptr=(int *)malloc(2*sizeof(int));
        ---------------------
        |     bss           | --> roba non inizializzata --> static int i;
        ---------------------
        |     data          | --> roba inizializzata --> int x=2; (in globale)
        ---------------------
        |     text          | --> programma vero e proprio
        ---------------------
```

LO STACK

LIFO -> Last In First Out
Usato O dal programmatore O dal compilatore
rispetta naturamente le chiamate di un programma

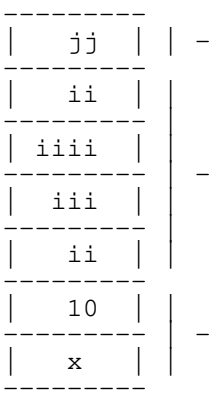
```
                |esempio: arrivo, in ordine, da main a callC
                '-----> main -> callA -> callB -> callC
                        main -> callA -> callB          - callC eseguita
                        main -> callA                   - callB eseguita
                        main                            - callA eseguita
```

I dati possono essere aggiunti in unità da multipli di 64b
La maggior parte delle CPU hanno istruzioni e registri specifici per la gestione dello STACK

```
                RUNTIME STACK
                        SS      stack segment
                        RSP     stack pointer <--- punta all'ultima operazione
                                                   che è occupata

                PUSH
                        inserisce una quad word sullo stack sottraendo 8 da RSP
                        e salvando il risultato in [RSP]
                        push reg/imm
                              |
                              '---> sub rsp, 8
                                    mov [rsp], reg/imm

                POP
                        legge una quad word da [RSP] e aggiunge 8 a RSP
                        POP reg/imm
                              |
                              '---> mov reg/mem, [RSP]
                                    add RSP, 8

        SCRIVERE UN PROGRAMMA ASSEMBLER CHE INVERTE IL CONTENUTO DI UNA STRINGA
                        ESEMPIO
                        INPUT:  CIAO
                        OUTPUT: OAIC
```

LO STACK E LE CHIAMATE DI FUNZIONE

Lo stack contiene i function pointer --> si allocano sia le funzioni sia i parametri passati ad esse <-- l'insieme di queste si dice STACK FRAME



Contiene inoltre gli INDIRIZZI DI RITORNO dalle funzioni (prima che ne venga chiamata una salva si salva sullo stack l'indirizzo della istruzione successiva ala funzione attuale)

```
CALL NOMESOTTOROUTINE --> push rip
                          jmp  nomesottoroutine

RET                   --> pop rip
```

PASSARE I PRAMETRI

Fino a 6 argometi sui registri
Gli altri sullo stack --> di norma lo fa il compilatore

USARE RBP

Prima dell'inizio della funzione il compilatore inserisce due/tre istruzioni dette PROLOGO

```
push    rpb
mov     rbp,rsp
(sub Local_bytes, %esp) --> opzionale (tiene
                            spazio per var
                            locali)
```

Prima del return il compilatore aggiunge tre istruzino dette EPILOGO

```
movl    rsp,rbp
pop     rbp
ret
```

A COSA SERVE IL PROLOGO?
serve a fare in modo che RSP possa cambiare senza modificare RBP inizale prima che questo venga modificato dalla sotto-routine

A COSA SERVE L'EPILOGO?
serve a annullare il prologo riportando lo stack alla situzaione precedente alla chiamata

USARE COOKIE.C per vincere -->
scambiare cookie e buf[80] (cookie in alto)
gcc -fno-stack-protector -o cookie cookie.c -->
-fno-stack-protector serve a disabilitare le protezioni MODERNE al buffer overflow
objdump -M intel -d cookie --> *PER CAPIRE QUANTO SPAZIO E' ALLOCATO a BUF*
./cookie
in input una dimensione adeguata di caratteri a caso e in fondo 0x41424344 (ABCD) IN LITTLE ENDIAN (DCBA)

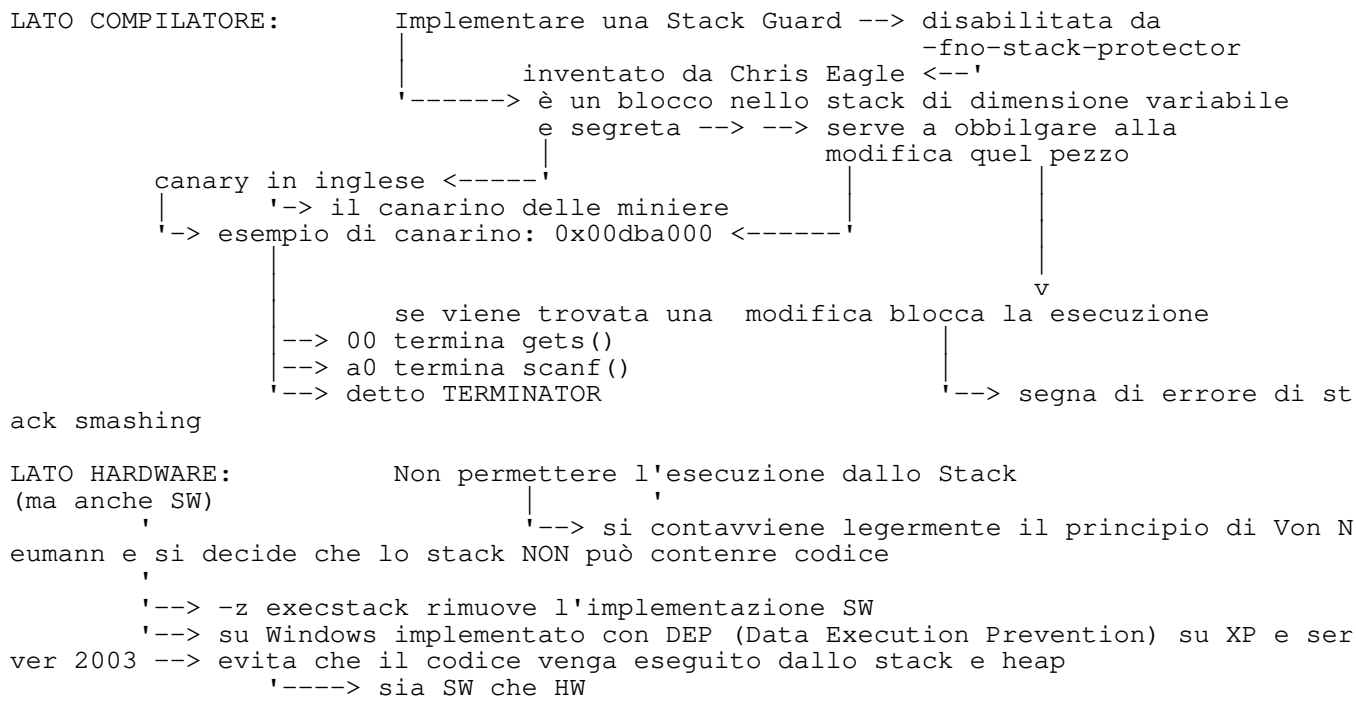
A CASA
foto (cookie modificato con 0x01020305 invece di 0x41424344) --> basta copiare il metodo precedente cambiando la parte di appiccico finale

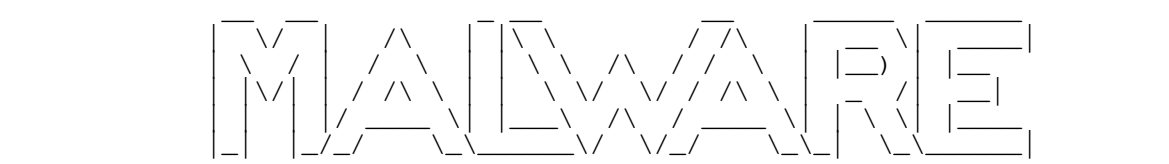
STRATEGIA DI ATTACCO

INIETTARE:
come le iniezioni in aula

```
                shell code scritto
GIA' PRONTO:


--> devo fare l'hijack del flusso di controllo
        |                       FOTO/retHijack.c scopri quale xx e modifica +8 a
        |                       +16/24/32 (scopri) e fai funzionare (value of
        |                       x=13)
        |                       --> FATTO vedi C/returnAddressHijack/rethijack.c
        |                       --> +4 al primo e +16 al secondo fa saltare da
        |                           c=function() a printf(...) senza passare
        |                           per x=10
        |
        |
        |
        v
uso gdb con vari breakpoint per studiare l'evoluzione dello stack nei vari
stadi e capire quanto devo spostare considerando che la printf in assembly
occupa diverse linee di mov e una lea, mentre x=10 occupa una sola movl
--> noi dobbiamo saltare alla istruzione subito dopo skippando una mov post
    call e la movl stessa

COMANDI UTILI
        COMPILAZIONE
                gcc -fno-stack-protection -z execstack fileSorg -o eseguibile
        DIABILITARE ASLR
                echo 0 > /proc/sys/kernel/randomize_va_space
                sudo sysctl -k kernel.randomize_va_space=0
        ENABLE CORE DUMP
                unlimit -c unlimited
        VEDERE CORE DUMP
                gdb -q exectable core

NUOVO PROGRAMMA cookie ma con prima buf e poi cookie (come originale)
                ricorda che
                                RA
                                RBP      probabilmente come per esercizio
                                         precedente (retHijack) con studio di
                                         stack
                                buffer                    ^
                                cookie                    |
                                                          |
devo saltare l'if sbufferando fino a ra <------ sbufferare non va
(con a ra l'indirizzo della print
nell'if IN LITTLE ENDIAN) vedi script
python in foto (buf+=.... con address vero)

                        *INJECTION CON CAMBIO DELLA RET*

Fare una iniezione di shell su un file SENZA debolezze (gravi, c'è una gets
comunque) modificando la ret alla fine del main per saltare ad un indirizzo di
stack dove abbiamo caricato il nostro codice.
vedi README in Injection/Lez2/ShellConRetMod per la procedura --> l'indirizzo
può variare a seconda dello stato del PC in quel momento, ATTENZIONE

PER CASA --> injection di shell su un programma stile vittima ma con un buffer
             di 4 !!
                        *CONTROMISURE AI MEMORY ERROR*

LATO SVILUPPATORE:      Non usare funzioni vulnerabili tipo gets() e scanf() ma
                        funzioni come strncpy,strncat,... che sono SICURE (la N)
                        |------> NON far copiare più B di quanti devono
                        '------> PROBLEMA PER SOFTWARE LEGACY (vecchio)
LATO SO:                Randomizzare il layout dell'address space --> ASLR
                        |------> risolve per il software legacy su sistemi
                        |        MODERNI
                        '------> rende molto difficile trovare i RA
                                         |
                                         '------> ci sono metodi per superare
                                                  ma sono molto complessi
                                                         |
                                                         v
si può scoprire tramite programmino che stampa l'indirizzo in HEAP e STACK di
```

due variabili





NIST 800-83 --> un programma diffuso su un sistema, di norma di nascosto, con
l'intento di compromettere la confidenzialità, integrità o
disponibilità del sistema vittima

Diversi tipi --> Worm, Trojan, Backdoor, Keylogger, Logic Bomb, ...

COMPOSIZIONDE DI UN MALWARE

```
-------------------------
| Propagazione | Payload | --> COSA FA
-------------------------          |
                                    |
                                    '---> COME SI MUOVE
```

CLASSIFICAZIONE

Possono essere divisi in diversi modi

```
|
|----------------> Come si diffonde                                  |----> MODO 1
|----------------> Cosa fanno una volta che hanno infettato          |
|
'----------------> Se hanno bisogno di un sistema ospite            |
        |              VIRUS                                        |
        |--------> Se sono "autonomi" (hanno tutto ciò che          |
        |          può servire al loro funzionamento)               |
        |          WORM, TROJAN, BOTS                               |----> MODO 2
        |--------> Se si replicano                                  |
        |                    WORM                                   |
        '--------> Se non si replicano                              |
                             TROJAN                                 |
```

PROLIFERAZIONE

Infezione di altro software
Sfruttamento di un exploit
Social Engeneering per rimuovere feature di sicurezza

AZIONI DEL PAYLOAD

Nascondersi
Compromettere file
Rubare informazioni
Distruggere la macchina

KIT DI ATTACCO

All'inizio dovevano essere precisissimi e specifici
Poi sono nati dei META-MALWARE --> enerano il codice malevolo a partire da poche
                | informazioni
                |
                |----> Zeus e Angler
                '----> Comprabili da soggetti poco raccomandabili

CHI LO FA

* Chi ha motivazioni politiche --> ?
* Criminali --> pochi
* Organizzazioni Criminali --> mafia?
* Organizzazioni che vendono al miglior offerente --> data brokers
* Intelligence --> CIA, NSA, FSB, Mossad, ...
* Smanettoni --> rarissimi ormai

Grossi movimenti di soldi, molto difficile ormai

ADVANCED PERSISTEN THREAT

Avanzato e persistente, come da nome
|--------------> usato dall'intelligence
|--------------> come prima cosa si nasconde |---> esempio SOLAR WINDS (1k anni umani di sviluppo, sgamato dopo 10 mesi dopo un controllo casuale sul traffico di rete)
'--------------> mandato a bersagli PRECISI

Tecniche specifiche di attacco e di mimesi

VIRUS

Programmi che INFETTANO un altro programma --> appiccicandosi al programma ospite (intero alla fine/inizio o a pezzi in giro)
'-----> sposta l'entry point al pezzo di virus e poi all'inizio dell'ospite --> file ELF
'-----> può fare tutto ciò che può fare l'ospite --> se root sono guai
'---> può provare a fare Priviledge Escalation

HEADER

PROGRAM

COME INFETTANO --> infection vector
QUANDO SI ATTIVANO --> trigger
COSA FANNO --> payload

FASI
IDLE/DORMIENTE --> non fa nulla finchè non c'è un trigger
--> non sempre presente
ATTIVAZIONE --> il trigger avviene
PROPAGAZIONE --> si riproduce
ESECUZIONE --> esegue il payload

MACRO VIRUS
Si attaccano a file vari (PDF, DOCX, immagini, ...) che sono scritti in diversi linguaggi --> Office in VisualBasic
Il virus è scritto nello stesso linguaggio ed è contenuto in questi file
Più semplici da scrivere e indipendenti dalla piattaforma

MELISSA --> primo macro virus
|---> infettava tutti i futuri documenti aperti con office
|---> mandava se stesso a 50 indirizzi su Outlook
|---> si segnava sulla macchina
'---> si attivava se il minuto era = all'ora

Una volta venivano identificati dalla SIGNATURE --> stringa UNIVOCA nel codice, quando trovata veniva aggiunta alla lista
'-->ORA NON PIU' --> sono stati sviluppati VIRUS CIFRATI
v
prima c'è il virus criptato POI la routine di decriptazione e encriptazione
v
sgamabile attarverso la routine che diventa la signature
v
evitabile attraverso la modifica (ogni volta) della routine --> la signature

```
                                        |                       cambia ogni
                                        |                       volta
                                        |       VIRUS POLIMORFICO <-'
                                        v
                                        Kaspersky capisce che POST decriptazione
                                        il virus è in CHIARO
                                        |       '--> lo si becca adesso in una
                                        |            VM che l'antivirus apre
                                        |            apposta
                                        |               '--> ANALISI STATICA
                                        v
                evitabile facendo in modo che il virus si modifichi da solo di
                generazione in generazione
                | '--> VIRUS METAMORFICO
                v
        sgamabile in esecuzione per vedere cosa fa --> se fa cose sospette lo
                                                       flaggo
                                                        '-> ANALISI DINAMICA

CLASSIFICAZIONE
        DIPENDENTE DAL BERASGLIO
                BOOT SECTOR --> zona dove c'è il kernel e che lo carica
                            '-> infetta OGNI volta che viene acceso/riavviato
                ESEGUIBILI  --> si appiccica all'eseguibile
                            '-> infetta al lancio
                DOCUMENTI   --> "    "       a un file --> infetta all'apertura
                SPARSO      --> mix di quelli sopra
        DIPENDENTE DALLA MIMESI
        a seconda di come si nasconde lo si classifica

                                        *WORM*

Malware AUTONOMO
Sfrutto problemi esterni (exploit) per farlo eseguire/proliferare
|--> ad esempio il SERVIZIO DI RETE
|                               '---> ascolta il traffico delle porte e risponde
|                                     di conseguenza
'--> spesso si manda in giro sulla rete A CASO
                |
                |
                v
                può farlo attraverso svariati modi
                                |---> email
                                |---> login
                                '---> app di messaggistica

In generale un Worm attacca una macchina (che ha un certo IP) e da qui si
propaga in diversi modi
                |---> cerca sistemi collegati
                '---> genera un IP a caso (o semi caso)

La crescita è inizialmente lenta, poi veloce e di nuovo lento a ciclo

        |                    .
        |                  .
        |                .
        |              :
        |            :
        |          .
        |        .
        |      .
        |    .
        |..::_________________

Il primo Worm è INTERNET WORM, altri sono MELISSA, CODE RED, STUXNET

                                *RAMSONWARE*

Criminalità organizzata, chiede un ricatto per riavere i dati, che vengono
criptate fino al pagamento (o anche oltre)
                        '---> scoperti dopo segnalazioni da utenti O sonde
                              automatiche
```

Ad esempio WANNACRY, che chiedeva 300$ in BTC per dare la chiave di cifratura
                        |
                        '----> fermato perché pingava un sito web e se questo
                               non rispondeva si attivava <-- KILL SWITCH

MOBILE CODE WORMS

Malware eseguibile da piattaforme diverse
Spesso scritto in Java/Javascript/VBScript

Il primo fu Cabir nel 2004, lo seguirono Lasco e CommWarrior nel 2005
Usavano Bluetooth e MMS, ORA usano i Marketplace (specialmente su Android)

DRIVE-BY-DOWNLOADS

Viene inividuata una vulnerabilità su un Browser, si fa scaricare (al momento della connessione ad un server) un documento che si autoesegue e sfrutta
la vulnerabilità del browser per replicarsi

CLICK-JACKING

Banner che spingono al click per infettare ("HAI VINTO UN IPHONE", "ALLARGA IL PENE", ...) e fare cose (dall'innocuo cancellarti le mail a cose più serie)

SOCIAL ENGENEERING

Fregare l'untente convincendolo ad aiutare inconsapevolmente la sua infezione
        '--> tipo gli scammer

COSA PUO' FARE IL PAYLOAD

DISTRUZIONE DEL SISTEMA
        Stuxnet --> rompeva le centrifughe
        Chernobyl Virus --> cancella il boot sector (Win 95/98)

PROPAGAZIONE DI WORM
Internet Worms --> vedi sopra

RANSOMWARE
Criptazione dell'harddisk e decriptazione attraverso riscatto
'--> dall'arrivo delle criptovalute sono popolarissimi e efficaci
        Wannacry --> vedi sopra

CREAZIONE DI BOTNET
Prendere il controllo del sistema e usarlo per i miei scopi
        '--> un migliaio conosciute
                                |--> DDoS
                                |--> Keylogger
                                |--> Cryptominer
                                |--> invio di malware
                                |--> spam
                                '--> controllo del traffico delle informazioni
                                     |--> questionari, statistiche, ...
                                     |
                                     |--> SCONFIGGIBILE "tagliando la testa" al
                                     | principale, ma è difficile e poco
                                     | utile
                                     |--> facilmente sostituito
                                     |
                                     '----> la REMOTE CONTROL
                                            FACILITY/COMMAND&CONTROL Network
                                            '-> il server per la nostra
                                                legione

SPYWARE
Controlla ogni azione che avviene sulla macchina --> spesso su cellulari
'--> jailbreak e ottiene i diritti di root
        '--> intercetta telefonate |
        '--> intercetta messaggi |---> e le invia a chi di dovere
        '--> controlla contatti |
        '--> tiene sotto controllo le immagini |

Molto più facilmente creabili per Android ---> funzionava anche su Apple

```
Il più famoso era PEGASUS della NSO, usato spesso dalle agenzie d'intelligence
        '--> ora deprecato UFFICIALMENTE ;-)

PHISHING
Rubare in maniera fraudolenta le informazioni di qualcuno attraverso inganni

BACKDOOR
        Punto di ingresso nascosto nella macchina ad uso futuro creato DURANTE la cr
eazione O aggiunto in seguito -> ad esempio il servizio di rete
        Usato anche per scopi legittimi (debugging)

ROOTKIT
        Una serie di programmi che vengono installati per dare all'attaccante i perm
essi di root
        Di varia natura
                PERSISTENTI --> a ogni boot in memoria
                AD ATTIVAZIONE --> si attivano al lancio del programma ospite
                                   e allo spegnimento si annulla
                USER MODE --> meno pericoloso
                KERNEL MODEc --> più pericoloso
                VM BASED --> sostituiscono il sistema con una esatta copia in
                             VM --> "letale"
        Ad esempio cambiano le syscall
        MOLTO COMPLESSO

                                *COME E' STATO CONTRASTATO NEGLI ANNI*

Idealmente la difesa da una QUALUNQUE minaccia è divisa in tre fasi
        PREVENZIONE
        IDENTIFICAZIONE
        RIMOZIONE

In IT Security la difesa si dvide in
        POLICY --> linee guida per l'uso (solo gli amministratori possono
                   fare X, ...)
        AWARENESS --> rendere consapevoli gli utenti (attenzione a leggere
                      l'indirizzo mail del mittente, ...)
        VULNERABILITY MITIGATION --> ridurre al minimo le vulnerabilità
        THREAT MITIGATION --> ridurre le minacce al sistema
        DETECTION --> sgamare la minaccia
        REMOVAL --> eliminare la minaccia


                                *GLI ANTIVIRUS*

Antivirus
| |---->4 GENERAZIONI
|               1a --> controllo della signature --> funzionava solo a
|                      "infezione nota"  (vedi sopra per dettagli)
|               2a --> congelava il SW sulla macchina e creare una signature
|                      per CIASCUN programma --> INTEGRITY CHECKING
|                                          '
|                                          '-> se il software cambiava essa
|                                              cambiava --> allarme di potenziale
|                                                           minaccia
|               3a --> esecuzione controllata del malware per analisi (vedi
|                      sopra per dettagli)
|               4a --> simulazione e confronto attraverso Machine Learning
|                  '-> non sono SOLO passivi, ma anche ATTIVI <-- Detection in
|                                                                 semi real time
|
|
| |---> possono creare FALSI POSITIVI
|
| |--->Funzionano o a SIGNATURE BASED o a BEHAVIOUR BASED
|       |--> in realtà in entrambi ormai perché gira tutt'ora il malware vecchio
|       '-> stringa univoca   '-> a seconda delle syscall (tra le altre cose)
|
|                                *LE SIGNATURE*
|
| |---> Segreto industriale dei produttori --> qui spiegato la minor o miglior
```

```
| |     difesa --> dati raccolti O da sensori sulla rete O sulle macchine degli
| |                utenti
| |                     '--> raccolti e analizzati in automatico
| |---> Per ridurre la concorrenza esiste il CME -> Common Signature Enumeration
| |                     |---> di MITRE e finanziato dagli USA
| |                     '---> mira a dare un identificatore unico e comune a
| |                           ogni malware
| |
| '---> Virus Total, di Google, analizza i file degli utenti che glielo
|       forniscono usando diversi antivirus ONLINE e pseudo-gratuitamente
v
ORA SONO CHIAMATI EDR --> Endpoint Detection & Recovery                    |
        '-> Analizzzando in automatico con Machine Learning, AI e          |
            Threat Intelligence                                            |-,
                     '-> analisi                                           | |
                                                                           |
                detti Next Generation Antiviruses <------------------------'

                                    *ABRA WORM*

Presentato a Purdue Uni
Worm --> infetta una e tenta di infettare altre
In particolare cerca di rubare dei file

        SCHEMA                              ssh           ssh
                                attaccante ---> INTERNET ---> vittima
                                        ^                     | |
                                        '---------------------- |
                                                               v
                                                             server

Prova ad usare ssh per connettersi con una bruteforce tra le password nel suo
dizionario
Deve rubare dei file
        |-->TUTTI quelli che contengono la parola "abracadara"
        '-->fa ls e grep per trovare i file
Esfiltra il codice (sia sul PC per debug, sia sul server) usando SCP (copia
tramite ssh)
Tenta di infettare le altre macchine
        '--> copia se stesso nelle macchine infettate e da loro tenta di
             trovare altri

Tutto su doker (VM)
'--> vedi guida qui:  https://git.laser.di.unimi.it/teozoia/abraworm
        VITTIMA usa la 2222 sulla 12345
        ATTACCANTE usa la 2222 sulla 22
eventualmente eliminare con
        rm -r ssh* --> per eliminare i file ssh
su un terminale
        sudo docker-compose up --force-recreate

su un altro terminale
        cd worm
        python3 AbraWorm.py

        FARE RAMSON WARE con libreria crypto? --> TBD
```

Nuove tecnice diverse da quelle "in locale" viste precedentemente
L'ambientazione è la stessa
|
v
:-) <--> [sistema] <--> >:-)
ma con più/diverse componenti nel sistema

COME FUNZIONA UNA RETE

ENDPOINT --> oggetto che riceve/host
PLUMBING --> link
Router/Switch --> fanno passare le info reindirizzandole tra link
Una rete è in insieme di calcolatri interconnessi per facilitare lo scambio di informazioni
Gli host comunicano attraverso messaggi, codificati come segnali elettrici attraverso passaggi fisici (cavi, onde)

Computer --> inviano e ricevono
Routers --> inoltrano pacchetti
Canali --> vie di passaggio

LAN (Local Area Network) --> rete locale (entro 1 km circa)
Tutti connessi a tutti --> una volta attraverso gli HUB --> fa un broadcast a tutti i computer della rete e solo il destinatario lo apre
'-> ora attraverso le SWITCH --> invia solo al destinatario

MAN (Metropolitan Area Network) --> rete a dimensione cittadina

WAN (Wide Area Network) --> rete a grande distanza --> Internet ad esempio
'-> insieme di LAN e MAN gestiti da infrastruttura di instradamento

Ogni computer si collega alla rete con una Network Interface Card --> due tipi, spesso combinati --> Ethernet
| '-> WiFi
|
v
configurabile in diversi modi, per esempio Promiscua
Ogni computer sulla rete è identificato da DUE componenti
|---> MAC --> univoco AL MONDO e legato alla scheda di rete
| |--> 48b (6 ottetti) --> i primi 3 sono forniti da un ente di certificazione (dipendenti dal manufactorer)
| |--> modificabile ma non fatto di norma
| '--> visibile con ifconfig
|
'---> IP ,-> univoco NELLA RETE LOCALE (una volta al Mondo) e generato da SW (dinamicamente)
|-> IPv4 32b (di norma) divisi in 4B dai valori da 0 a 255 separati da .
|-> IPv6 128b divisi in 8 gruppi da 2B ciascuno in esadecimale separati da :
'--> normalmente è dinamico, solo alcuni lo hanno statico

```
SWITCHING
        |--> Circuit --> come il vecchio sistema telefononico
        |             '-> unico circuito con scambi fisici su dispositivi
        |             '-> circuito mantenuto fino alla fine della trasmissione
        |                          '--> percorso unico
        |
        '--> Packet ,-> dati divisi in pacchetti --> 1500B ciascuno in media
                    |-> pacchetti inviati sulla rete in maniera indipendente
                    |   '--> anche in maniera disordinata, vengono riordinati
                    |        dal ricevente
                    |-> mandati cercando di avere l'efficienza maggiore
                    '-> l'importante è il punto di inizio e di fine, non il
                        percorso

PROTOCOLLI
        |--> Connectionless --> mandano i dati appena ne ha abbastanza da
        |                       mandare --> UDP
        |--> Connection-Oriented ,-> si occupa prima di mandare di creare una
        |                        |   connessione stabile e affidabile
        |                        |    '--> simula una commutazione a circuito
        |                        |-> dopo avere creato la connessione invia
        |                        |    finta
        |                        '-> dopo l'invio chiude la connessione --> TCP
        |
        '--> i 7 livelli OSI e i 4 segmenti di TCP/IP
                |                              |
                v                              v
                APPLICAZIONE                   APPLICAZIONE
                PRESENTAZIONE                  TRASPORTO
                SESSIONE                       INTERNET
                TRASPORTO                      INTERFACCIA DI RETE
                NETWORK
                DATA LINK
                FISICO

ARP e ARC
        ARP request vuole mandare una cosa --> chiede indirizzo, chi lo a
        risponde e poi si invia
                    '-> se nessuno risponde NON manda su LAN ma su Internet

                                *OBIETTIVO PRIMARIO*

Il canale di comunicazione può essere usato come nuovo modo di violazione, così
come router/switch (utili per sniffare il traffico) e protocolli (SSH,HTTP, ...)
La rete può essere usata per propagare il Malware

Sfruttano principalmente i problemi della implementazione della rete

                                *ATTACCHI PASSIVI*

Intercettazione di dati e analisi del traffico
Insidioso e difficile da scoprire

                                *ATTACCHI ATTIVI*

Modifica/falsificazione di dati passati e Denial Of Service (DOS)

                                *SPOOFING*

Interpretazione
Modifica dell'indirizo sorgente di un pacchetto per assumere l'identità di
diverse cose come:
        * un server
        * un router
        * un utente
        * un computer
        * un servizio web
                + un sito
                + un servizio mail
        * un servizo gps

I sistemi ORIGINARIAMENTE non fornivano un servizio di autenticazione (anche ora
è raro)
```

Maschero il mio ip con un altro (potenzialmente con quello di un altro)
IPv4 lo permette

PACKET SNIFFING

Intercettare il traffico di rete
facile su doppino, ethernet e onda radio
più complesso su fibra

Su LAN su hub è semplice --> si può rendere la nostra scheda di rete PROMISCUA
intercetta TUTTI i pacchetti, anzichè solo quelli per lei <--'
Su LAN con switch è più complesso
Su LAN a onde radio (WiFi) basta una antenna di ricezione e una scheda PROMISCUA
Ancora più facile con satelliti

Su Fibra bisogna intercettare il segnale luminoso, attaccarsi direttamente al cavo

Ormai il traffico è crittografato, quindi l'intercettazione non è più così facile/utile

Esistono KIT SNIFFER (tipo quelli da malware) per sniffare e analizzare i pacchetti su una rete --> Wireshark
'-> sia per scopi benevoli
'-> sia per scopi malevoli

Difficilmente scopribile in quanto attacco PASSIVO

Evitabile attraverso la criptazione dei pacchetti

FINGERPRINTING

Prima fase di un attacco, è ricognizione
Sfrutta i protocolli di rete --> fa l'impronta di un sistema per poterlo riconoscere (scoprire tutte le informazioni) e da lì analizzarlo

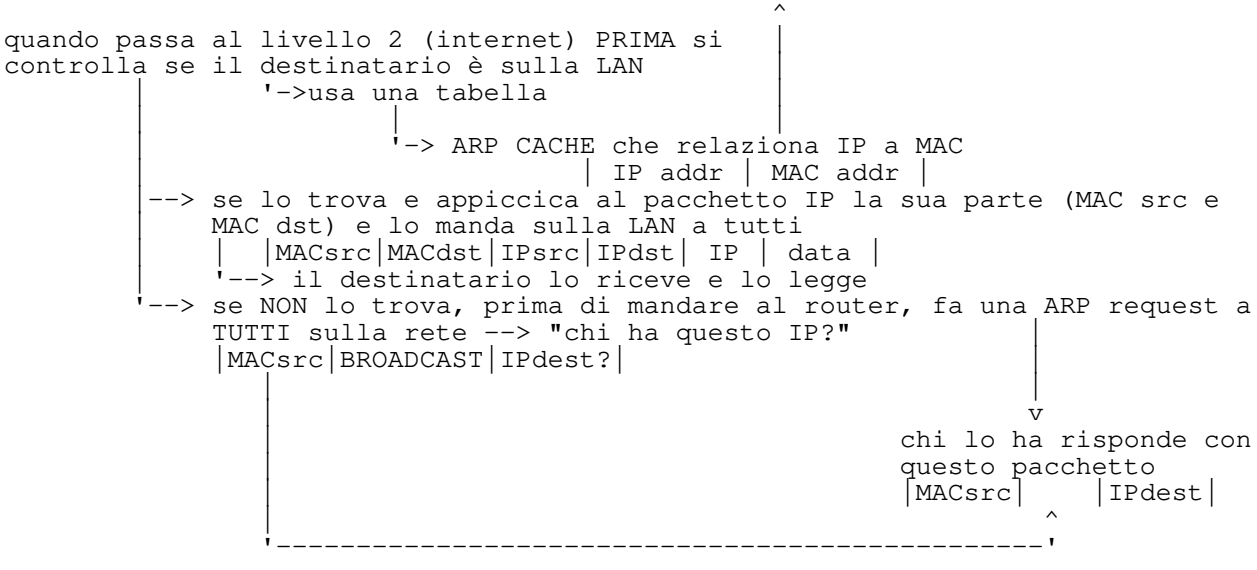
PORT SCANNING

Sondare le porte di rete di un sistema mandando pacchetti a ciascuna per analizzare la loro risposta (se sono aperte, che servizi ospitano, ...)
Dopo avere scoperto i servizi aperti si cercano (su CVE) le vulnerabilità di ciascuno

Si può fare con nmap, ma ormai si sa come funziona --> difficile farlo ormai, troppe richieste fanno partire l'allarme

PROTOCOLLO ARP

Pacchetto ip (livello 3):
|IPsrc|IPdst| IP | data |
le entry durano pochi secondi



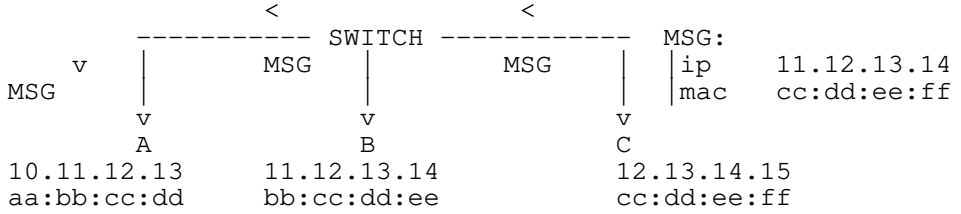
MITM: MAN IN THE MIDDLE

L'attaccante si mette in mezzo tra due host e intercetta le informazioni scambiate da essi, arrivando anche a modificre le stesse
Sfrutta varie cose --> ARP CACHE poisoning <-- vediamo questo
'-> DNS spoofing
'-> SSL hijacking

ARP CACHE POISONING

Ogni volta che ARP riceve un pacchetto essa si salva l'IP e MAC per tenere aggiornate le entry (che durano poco, vedi sopra)
'-> evita di dover fare continue richieste ARP
E' stateless, non si ricorda delle richieste --> se riceve una risposta "si fida" di avere fatto una domanda a riguardo e che il dato sia giusto
|
'--> può essere fregato mandando risposte a richieste NON eseguite
'--> si può dire di essere una macchina con diverso IP mandando una reply con l'IP da sostituire e il proprio MAC
Mandiamo risposte senza avere ricevuto richieste --> GRATUITOUS REQUESTS

HALF-MITM con ARP POISONING

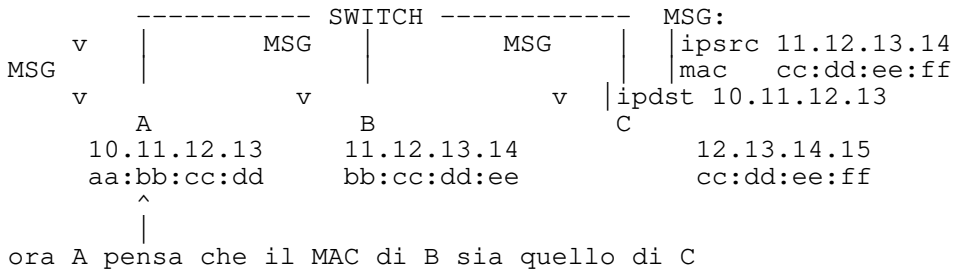


ora A pensa che il mac di B sia quello di C e manda a C il traffico da A a B
Da aggiornare periodicamente durante l'attacco (mediamente ogni 40 secondi)
E' HALF perchè intercettiamo solo ciò che esce da A verso B, non anche da B ad A

FULL MITM

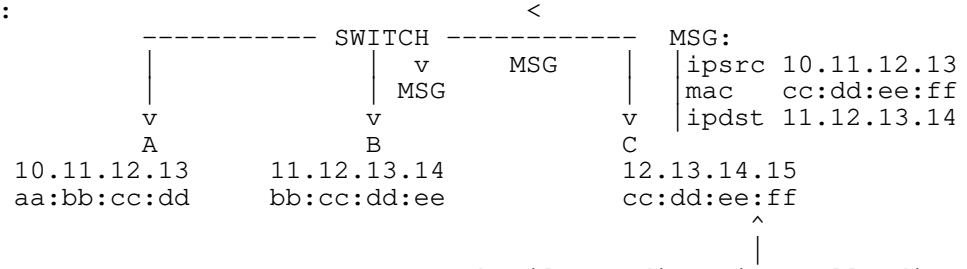
Intecetta sia da A a B sia da B a A
Manda il doppio dei pacchetti ARP
Ettercap ad esempio è un tool che lo fa

PARTE 1:



ora A pensa che il MAC di B sia quello di C

PARTE 2:



ora B pensa che il MAC di A sia quello di C

Con le due parti C riceve il traffico completo tra A e B

Ci si protegge con entry statiche --> casi rari e particolari come per esempio una Caserma

TCP HIJACKING

Vedi sotto come funziona il TCP (SYN e ACK)

Facciamo in modo che l'utente chieda A NOI il SYN, che lo chiediamo al server, che riponde a noi, che rispondiamo all'utente che risponde a noi che rispondiamo al server --> ci mettiamo in mezzo come tramite
Devo sapere il valore di X e di Y
'-> complesso perché è generato dal server e client

Come si capiscono X e Y?
'--> si sniffa il traffico
'--> si indovina

Il primo lo fece (si suppone, ma lui nega) Mitnick

L'ATTACCO DI MITNICK

Comando rsh (Remote Shell)
rsh ip_address shell_command --> se si omette shell_command ci si connette con rlogin
NON LO PUO' FARE CON TUTTI
c'è un sistema di login
--> ai tempi di Mitnick c'erano .rhosts e /etc/hosts.equiv
'-> ogni volta che si effettuava una connessione si ' controllava che fosse segnato qui l'hostname
'-> se non c'era in /etc/hosts.equiv si cercava su .rhosts
Shimomura (da ora in poi S) aveva creato una connessione accreditata (su hosts.equiv) tra il suo computer e quello della NSA (d'ora in poi A)
accreditata
S ----------> A

Mitnick (da ora in poi M) esegue
1 finger indirizzo_S
2 shmount -e indirizzo_S (elenca tutte le macchine con connessioni accreditate)
3 manda una 20ina di SYN alla macchina di S --> lei risponde con altrettanti SYN/ACK
--> M vuole capire in che modo viene creato Y
'-> capisce che il +1 è +400 --> capisce l'algoritmo
4 fa una SYN-flood su A e la mette fuori uso (lo tiene attivo)
5 fa un nuovo SYN a S fingendosi A sulla porta 53
|IP_A|IP_S|53| X |...|
--> S fa SYN Y/ACK X+1 a A
|IP_S|IP_A|53|X+1| Y |
--> A dovrebbe mandare un ACK Y+1 ma non può
--> M manda un ACK con Y+1 calcolato dal passo 3
|IP_A|IP_S|53|Y+1|...|
6 S apre la connessione a M come se fosse A
7 M esegue 'echo ++ > ~/rhosts' --> mette ++ in rhost vuole dire che CHIUNQUE è affidabile <--'
8 chiude la SYN-flood su A, che riprende a funzionare
9 ruba dei documenti MA non cancella i log --> S lo sgama, si allea con l'FBI e dopo un mese lo catturano
'-> 5 anni di carcere e divieto di uso di Internet per altri 5

Subito dopo l'attacco ne avvennero molti altri finché il protocollo il TCP
ORA molto difficile fare il punto 3 perché hanno migliorato gli algoritmi

DOS E DDOS

Denial Of Service e Distributed Denial Of Service --> vanno a minare la DISPONIBILITA' di un servizio --> non si riesce a difendere
'---> stessa cosa ma più macchine fanno l'attacco <-- SPESSO BOTNET --> record di traffico 2.3 Tb/s (2020)
--> blocca un utente legittimato a farlo dall'usare un servizio/sistem a sulla rete
'--> mira a far esaurire una risorsa su un sistema --> tempo di CPU,

memoria, BANDA, la macchina in se (Brain distruggeva la GPU)

Esempio in locale --> riempie la tabella dei processi del computer

```
while true{
        fork();
}
```

In locale si può riempire il FS, creare processi (su), uccidere processi, ...
Sulla rete si possoo mandare pacchetti malformati, una marea di pacchetti, smurf

Usato per sabotaggio

SMURF ATTACK

Si fa un enorme numero di PING alla macchina e le si blocca la connettività

Comodo fare così:
ATTACCANTE crea un pacchetto di PING ma modifica SRC mettendo l'indirizzo della vittima e come DST broadcast di una rete grossa e lo manda
Tutti gli host della rete ricevono un PING dalla vittima e le rispondono

la VITTIMA riceve un numero altissimo di ping bloccandola

SOLVED --> i router bloccano i pacchetti in broadcast

TCP DOS

Va a saturare la tabella delle connessioni TCP di una macchina

La connessione TCP funziona a SYN e ACK in THREE HAND-SHAKE

```
                SYN X
---             ---->       ---       X e Y sono generati a caso
HOST    '_'     <----       '_' SERVER
            ACK X+1 e SYN Y
                ---->
                ACK Y+1
```

Il server aspetta l'ACK Y+1 per finalizzare la connessione --> riconosce un utente in base a IP/MAC e y+1

Si continuano a fare SYN X al server, che risponde con SYN Y|ACK X+1, ma non si risponde MAI con ACK Y+1
Il server continua a creare nuove voci nella sua tabella fino a saturarla e non essere più disponibile

COSA E' UN BROWSER?

Programma che interpreta diversi file ipertestuali e li traduce in pagine più human readable e interfaccia l'utente a tutto ciò che è sotto
Spesso si usano linguaggi di markup (HTML, ...), abbellimento (CSS, ...), da database (SQL, ...) e di programmazione vera e propria (JavaScript, ...)

Può aggiustare un sorgente (solo per lui) per renderlo visibile e conforme allo standard HTML

Permette di vedere la sorgente e di ispezionare la pagina nella sua completezza

LA CONSOLE DEL BROWSER

Ci si interagisce con javascript
Si possono lanciare funzioni interne al documento o usare comandi generici

E' un interprete

Fa eseguire tutto lato client --> mai server --> il rendering è SOLO lato client, il server da' solo le impostazioni di BASE

PHP

Permette di lanciare, in locale, un Web Server semplice per lo sviluppo
Utile per vedere le risposte del server e confrontarle con le risposte del client

php -s 127.0.0.1:8000 apre un server IN LOCALE sulla macchina alla porta 8000

se usiamo netstat -tulpn possiamo vedere che c'è un server PHP "in ascolto" al localhost --> aspetta una connessione

PAGINE STATICHE

HTML normale su server (no interazione)

PAGINE DINAMICHE

Quando la parte server inserisce dati dinamici in base alla richiesta ad una pagina web

tipo il login "buongiorno GIACOMO"

Serve un linguaggio di BackEnd --> PHP

VULNERABILITA'

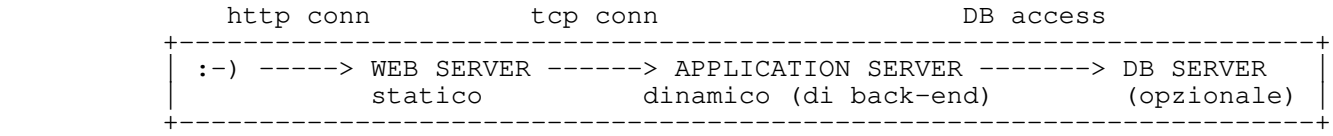
Possono esistere metodi di input NON controllati, vulnerabili alla Injection
'--> javascript ad esempio --> xss payload
Ci si può difendere facendo filtraggio dell'input --> ci sono librerie, non fare a mano

XSS REFLECTED

Creo un link speciale con il mio payload e lo mando all'utente
Possiamo estrarre i cookie, magari quelli di sessione
Possiamo inserire un keylogger

Vedi xss game appspot (tipo pwn/bandit)

SQL



Useremo PostgresSQL
Funziona a query

tipo: sqlRequest="SELECT * FROM Products WHERE Name CONTAINS '" + USER_INPUT + "' ";

cerca tra Products tutti quelli che contengono un certo nome specificato dall'utente

INJECTION

Modifica di una query attraverso input che dovrebbe essere "sano" --> tipo XSS

Esempi di injection

+ SELECT * FROM Products WHERE id=15; DROP TABLE Suppliers
cerca tra tutti gli id di Products quello che ha 15 E cancella la tabella Suppliers --> dopo E c'è la parte malevola aggiunta da qualcun'altro

Un altro esempio su ricerca da nome <-- injection più ez

' OR 1=1 -- --> fa chiudere il nome a '', mette una cosa sempre vera (il trick) e poi commenta il ' che ci deve essere nella query
^
'- commento in SQL

ORDER BY n per capire le colonne
UNION by _,_,_,... quante colonne

per capire i tipi suo NULL in tutti tranne uno e tento (prima numeri e caratteri, poi il resto)

Esiste una tabella che contiene i nomi di TUTTE le altre tabelle --> filtrare le cose utili da lì --> in PostgreSQL c'è

```
select table_name FROM information.schema
```

Si suppone ci sia il caso peggiore (mostra solo 1 riga), quindi si usa string_agg per aggregare OGNI stringa nella tabella in una unica separata da un separatore che si specifica nella funzione

E' possibile usare version() con una UNION SELECT a 3 (null, version(), null FROM ps_stat_user_tables) e --
versione modificata

```
...,pg_ls_dir('/bin'),...
```

E' possibile legegre file con pg_read_file

Payload SQL Injection
SQLmap automatizza

Esistono, come per html, dei metdodi di protezione (tipo HTMLSpecial...) e c'è anche il Logging (almeno tengo d'occhio passivamente tipo videocamere)

ATTACCHI ONLINE

Due VM che girano SOLO in locale:

* una con Kali (attaccante)
* una con Metasploitable 2 (vittima)

Da Kali con Wireshark seleziono eth0 per vedere tutti i pacchetti che passano dalla scheda (lanciando ping google.com vediamo i pacchetti di ping)
Traceroute elenca tutti i punti di passaggio di un pacchetto con ttl (time to live) progressivamente più basso

```
traceroute -n ip_address
```

'-> serve ad evitare che un pacchetto rimanga vivo all'infinito se si blocca

Per vedere TUTTI i servizi aperti uso nmap

```
nmap -sT  ip_address --> SOLO SU LOCALE (è pericoloso)
```

Come si è evoluta la protezione dagli attacchi? --> spesso più che non rendere possibile l'attacco rende inutile il risultato

3 momenti di OGNI meccanismo di protezione
--> Autentication --> chiedere chi è
--> Authorization --> dare l'autorizzazione a passare
--> Auditing --> decidere se le autorizzazioni sono legittime

Access Control
'-> <18 via, >= 18 dentro, VIP in area VIP in discoteca

Enforcment mechanism
mettere in atto le politiche di sicurezza--> muri, porte e infrastruttura della discoteca

Accountability
tenere traccia di tutto ciò che viene fatto e che le regole siano applicate --> videocamere in discoteca

Subjects --> entità attive --> fanno le operazioni --> utenti
Principals --> per alcuni uguali ai Subjects, per altri separati e conicidono con i processi
Objects --> entità passive --> subisocono le azioni dei Subjects --> memorie

Rights --> regole di accesso --> chi può fare cosa (si applicano ai Subjects quando si interfacciano con gli Objects)

Reference Monitor --> set di meccanismi che provvedono al controllo degli accessi.
* deve essere incorruttibile (Tamper Proof)
* è distribuito nel sistema
* deve essere PICCOLO --> meno cose che possono avere problemi, meno problemi

AUTHENTICATION

L'ID tra umani funziona face-to-face, ma per le macchine non è così, necessita di un meccanismo diverso --> trovato negli anni 60-70

è in due fasi
1) autocertificazione dell'utente
2) controllo della affermzione da parte dell'utente

1 è diviso in diverse tipologie a seconda delle organizzaioni (Nome e Cognome, stringa alfanumerica, mail, ...) --> USERNAME

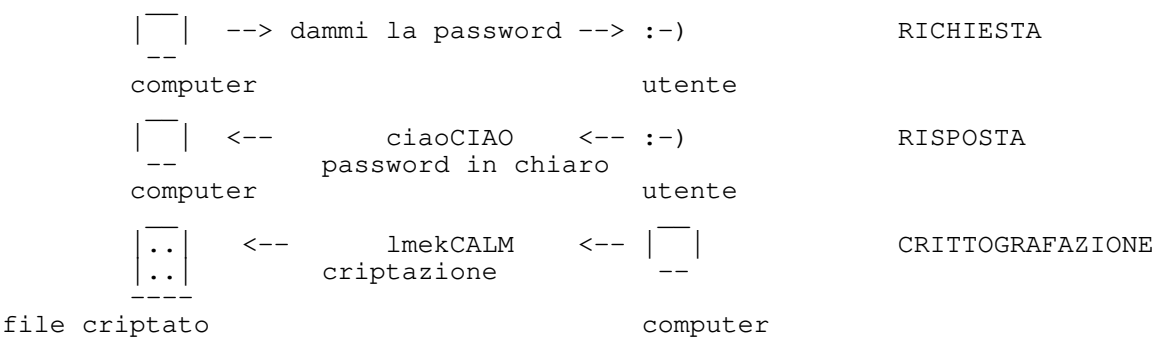
per eseguire la parte 2 sono state studiate negli anni 3

strategie

* WHAT YOU KNOW? il calcolatore ha condiviso un segreto con l'utente e ad ogni autenticazione l'utente DEVE ripeterlo al computer. Se lo sa allora può accedere
* WHAT YOU HAVE? il calcolatore sa che l'utente ha un oggetto UNIVOCO e lo conosce. Ad ogni autorizzazione l'utente lo deve ricondivide
* WHAT YOU ARE? il calcolatore conosce una caratteristica FISICA dell'utente, ad esempio una impronta digitale

WHAT YOU KNOW - approfondito

La password viene chiesta la prima volta all'utente e viene, dopo l'immissione, crittografata (in vari modi) e salvata come codice su un file



Il file che contiene le PW (in Linux etc/shadow) è accessibile solo da utenti con privilegi d sistema, che comunque, grazie alla crittazione NON possono leggere le password --> al massimo può resettarla

WHAT YOU KNOW - VULNERABILITA'

ATTACCHI LOW TECH
* Uso di password FACILI:
  * nome della squadra di calcio
  * nome della mamma
  * data di nascita
  * 1234...
  * qwertyuiop
  * ...

---> FATTORE UMANO rovina la sicurezza --> sistemi di suggerimento ("la password è troppo debole",...) possono arginare MA NON EVITARE

* Social Engeneering --> consocere la persona per prevedere la password
  '--> appasionato di cavalli che usa il nome della bestia come password

* Shoulder Surfing --> spiare l'utente per rubare la password

* Phishing --> falsi servizi che rubano le informazioni di accesso

ATTACCHI HIGH TECH
* Brute Force Attack --> tenta, basandosi su un alfabeto, tutte le possibili ----combinazioni delle lettere in una data lunghezza
  '-> se si ha accesso a etc/shadow si confronta (volta per volta) il risultato del BFA con le password listate
  '
  '
  '--> l'algoritmo di criptazione è noto (md5, ...)
  '-> la difficoltà è ESPONENZIALE (3h per 6 caratteri, 2 anni per 8 caratteri, ...)
  '
  '-> di norma si restringe il campo su parle ben definite (le x password più comuni, ...) che sono contenute in un dizionario reperibile su Internet
  |

```
* Dictionary Attack <-----------------------------'--> 20% di probabilità di
                  |                                   successo
                  |
                  '--> John the Ripper è un tool che permette di farlo
                       '--> nasce per aiutare i System Manager a tenere le
                            password degli utenti dei sistemi che gestiscono
                            siano sicure

* Rainbow Attack --> salta la parte di conversione da chiaro a criptato e usa
                     direttamente le parole hashate

                        PER PROTEGGERSI DAL RAINBOW SI USA IL *SALT*

                processo normale
                                          pwd in chiaro --> crittografia
                                          --> pwd crittografata

                con il sale
                                          numeri/caratteri casuale + pwd in chiaro
                                          --> crittografia --> pwd crittografata
                                                               SALATA

                        *WHAT YOU ARE - approfondito*

Tratti più usati:
        * impronte digitali
        * retina
        * viso
        * mano
        * voce
        * abitudini di battito
        * DNA (un po' estremo eh)

                        *WHAT YOU ARE - VULNERABILITA'*

* Può essere INTRUSIVA (lettura della retina avviene con un laser diretto
  nell'occhio)
* Può essere COSTOSO (analizzatore del DNA)
* Single Point of Faliure (se ho un taglio sul dito non posso usare il lettore)
* Può fare errori di lettura (sensore sporco)
* Lettura lenta (deve legegre e analizzare)
* Può essere contraffatta (mano mozzata, falso dito, maschera)
* Dati critici (sono MOLTO difficili da sostituire)
* Può dare falsi positivi o negativi (un errore in lettura può dare l'accesso a
  un non autorizzato o negare l'accesso ad un autorizzato)
                                          |
                                          '--> AUC dovrebbe essere 1 ma è 0.7

                        *WHAT YOU ARE - AUTENTICAZIONE*

ENROLLMENT:
        al primo avvio
        * legge la caratteristica in MODO PRECISO
        * individua caratteristiche specifiche
        * salva in bit
RECOGNITION:
        * legge la caratteristica
        * individua caratteristiche specifiche
        * le confronta con quello salvato
                + match
                + no match

                        *WHAT YOU HAVE - approfondito*

Usa un oggetto o attivo o passivo:
* OTP (attivo) --> prima su dispositivo poi su cellulare
* Smart Card tipo la CIE --> chip interno che fa crittografia
* SIM Card (Subscriber Identity Module)
        + contiene diverse informazioni (numero, nome, chiave segreta di
          autenticazione GSM)

                        *WHAT YOU HAVE - VULNERABILITA'*
```

Bancomat Skimming --> si rubano i dati attraverso un falso bancomat
Chiamata che punta a registrare e campionare la voce

MULTIFACTOR AUTENTICATION

WHAT YOU KNOW + di solito WHAT YOU HAVE (per esempio le banche usano WHAT YOU ARE)

SINGLE SIGN ON

Servizio che eprmette di usre un'unica password per più servizi
'--> GOOGLE (Gmail, Drive, YouTube...)

FEDERATED IDENTITY MANAGEMENT

Permette di usare lo stesso account su più organizzazioni.
Una delle organizzazioni fa da "testimone" per la mia identità al mio login
Ad esempio EDUROAM

OPEN ID

Permette che un ente sfrutti il servizio di autenticazione di qualcun'altro
Ad esempio Canva con Google

titolo host protection system

ACCES CONTROL

IN BREVE: * sistemi che prevedono il bloccare l'uso di una risorsa non autorizzata o l'uso non autorizzato di una risorsa
* serve una SECURITY POLICY
* è il cuore di un sistema di sicurezza
* implementato in diversi luoghi --> kernel, OS, applicazione
* Acces Control Matrix è lo strumento che garantisce cosa OGNI utente può fare --> vedi prima parte per la tabella

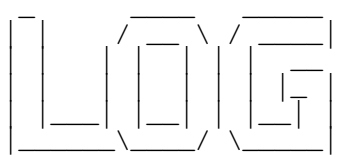
Diversi tipi:
* Discretional Acces Control - DAC
* Role Based Acces Control - RBAC
* ...
* Mandatory Acces Control - MAC

MAC ACCESS CONTROL

L'utente/applicazione NON deve essere in grado, indipendentemente dal suo livello, di modificare l'SO

Bisogna usare l'HW perché l'SO NON è in grado di proteggersi --> dovrebbe ogni volta che si fa un salto
oltretutto dovrebbe capire <---- controllare quando chiamare
'-> dovrebbe interrompere la applicazione --> quindi la CPU
!NON fermabile! <--'
v
usa un sistema MOLTO complesso in runtime:
PREAMBOLO
* si assegnano delle ETICHETTE sia a SOGGETTI sia a OGGETTI
* a seconda della etichetta la richiesta è esaudita o no
In INTEL ci sono 4 etichette/livelli/anelli concentrici:
+ 0 --> il + potente
+ 1
+ 2
+ 3 --> il - potente
privilegio
Per ora sui sistemi più comuni si usano SOLO 0 e 3
* ogni processo è diviso in segmenti che isolano i pezzi in soggetti e oggetti --> usata per evitare che due processi si infastidiscano
* ogni segmento ha un descrittore che ne elenca le caratteristiche e una etichetta (o 0 o 3 vedi sopra). I descrittori sono in una tabella che li elenca tutti
+ il Descrittore ha un campo chiamato DPL Descriptor Priviledge

```
          Level --> specifica la etichetta --> dettato da SO
        + esiste anche un CPL Current Priviledge Level che indica il
          PL del processo in esecuzione --> salvato su un registro
                                            quindi in HW
        + quando si cambia un processo il CPL diventa il DPL del nuovo
          processo

PROTEZIONE DEI DATI
Il codice utente NON può modificare i dati del SO
Un soggetto di livello X NON può accedere ai dati di livello inferiore,
un utente (livello 3) non può accedere ai dati dell'SO che sono livello
0, ma solo a quelli uguali o superiori al suo

Ogni volta che un processo vuole accedere a dei dati si controlla il
CPL con il DPL dei dati.
        * CPL<=DPL accesso garantito
        * CPL>DPL  accesso negato

Può accadere che un codice NON privilegiato chieda a del codice
privilegiato di accedere a dei dati privilegiati
|    B (lvl 3) --> "dimmi D" --> A(lvl 0) --> "cosa sei?" --> D (lvl 0)
v
PER EVITARE si usa RPL Request Priviledge Level che si segna il livello
della richiesta iniziale
        Prima di fare la richiesta si confrontano RPL e CPL (nel nostro
        caso 3 e 0), si prende il maggiore e poi si esegue la richiesta
     (MAX(DPL_B,CPL_A))<=DPL_D? NO --> MAX(3,0)=3 e 3>0
Esistono anche un campo chiamato Type che serve ad indicare che
operazioni possono essere eseguite
        * 0000 - Read Only
        * 0001 - Read Only,  Accessed
        * 0010 - Read Write, Accessed
        * 0011 - Read Write, Accessed

PROTEZIONE DEL CODICE
L'utente può usare SOLO ALCUNE parti di codice del SO

La politica di accesso è diverso dai dati:
        * un codice può essere eseguito solo se il
          CPL_pre_salto==DPL_dopo_salto
                Kernel salta solo su Kernel
                Utente salta solo su Utente
                + si evita di andare in zone non controllate dal
                  pre_salto
                + il codice a privilegi bassi può essere con errori
                + non ha senso togliersi privilegi

Come posso usare delle cose che solo il Kenel può fare?
'--> uso le CONTROLLED INVOCATIONS --> le syscall
     Permettono di saltare a zone a privilegio diverso attraverso delle
     "strade" ben definite --> una specie di visita organizzata
Si usano diversi meccanismi:
     * Call Gate --> syscall
     * Software Interrupt
     * SYSENTER/SYSEXIT
Il processo viene interroto, si salva il contesto del processo e si
esegue un JUMP ad un indirizzo preventivamente salvato su una tabella
che salva le zone "permesse" del Kernel --> che è in esecuzione senza
|       '-> Interrupt Descriptor Table      il rischio di venire
|           IDT                             disturbato dal software
|                                           utente, che è fermo
'-> Un attacco può modicare la tabella IDT per fare ciò che vuole
    Per esempio può modificare da lì il comportamento della scheda di
    rete per mandare sia al destinatario originale sia all'hacker il
    traffico

TUTTE queste tabelle hanno il loro indirizzo salvato solo su dei
registri che sono modificabili SOLO da livelli 0 --> gli altri generano
|                       '--> solo ROOT e Kernel      una eccezione dette
|                                                    General Protection
'--> un utente NON potrà mai modificarlo, tranne con notevoli sforzi
```



IT AUDIT

Esaminzaione e valutazione delle prestazioni di un sistema

IT SECURITY AUDIT

Valuta se la sicurezza della infrastruttura e dell'uso di quest'ultima rispetta il livello richiesto

COME FANNO?

Hanno accesso ai LOG di sistema, dove sono segnate TUTTE le operazioni significative eseguite da TUTTI gli utenti e processi in un sistema
Se una di queste operazioni devia dalla norma suona l'allarme

Non un lavoro facile e veloce
2 fasi
- * RACCOLTA DELLE INFORMAZIONI --> raccogliere i LOG
- * ANALISI DELLE INFORMAZIONI
  - + POST MORTEM --> dopo un attacco si capisce cosa è successo
  - + REAL TIME --> prima o durante l'attacco

COME SONO FATTI

Segnano tutto
Scritti dalla accensione allo spegnimento
Su Linux è su var/log/
- Diverse zone che segnano cose diverse (login, traffico internet, ...)
- Diversi formati a seconda della versione

Su un portatile quando il Log è pieno si sovrascrive --> circa ogni due settimane
Ogni log ha un codice di Severity da 0 a 7 che indica il pericolo, più basso è più è grave
Ogni log ha un codice di Facility che dice chi lo ha generato (su Linux da 0 a 8)
Sono file di testo (leggibili da classici lettori) o di formati proprietari che necessitano di programmi appositi
Per cercare nei log si usa grep --> ora però è difficile data la mole enorme di dati

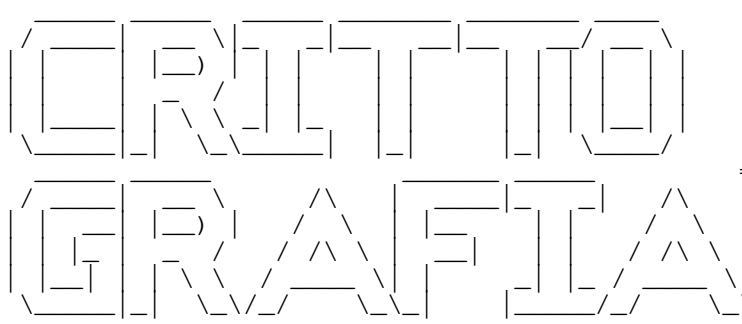
Data la MOLE esistono delle PIATTAFORME DI CENTRALIZZAZIONE che ricevono log da svariate macchine in diversi formati e li salvano su database SQL
- '-> rendono possibile la lettura tramite SQL anzichè svariati metodi

Esiste anche l'analisi automatica dei LOG tramite Machine Learning per analizzare senza l'ausilio di umani i log (ovviamente dopo training) e prevedere attacchi e vulnerablità
'--> chiamati LMS (Log Management Systems) o più modernamente SIEM (SECURITY, INFORMATION, EVENT MEANAGERS)

Su Windows c'è un programma di default per esaminare il log e definire una policy diversa

In generale PIU LOG si hanno più informazioni si possiedono MA PIU' SPAZIO è occupato

---



COSA E'

La scienza della occultazione revertibile delle informazioni attraverso delle trasformazioni MANTENDENDO l'informazione originale

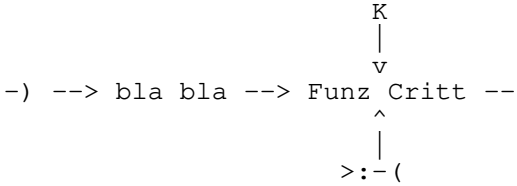
IL SUO CONTRARIO

CRITTOANALISI --> scienza che verte sulla reversione di un oggetto criptato

SI BASA SU

Confidenzialità

MODELLO DI RIFERMENTO



LA TEORIA DIETRO

La Criptazione è BIETTIVA (sia iniettiva sia suriettiva)
- OGNI elemento del dominio ha UNO e UNO solo corrispettivo nel codominio
- OGNI elemento del codominio ha UNO e UNO solo corrispettivo nel dominio

LE TECNICHE PRINCIPALI

Due tipi principali:
- quella storica era a CHIAVE PRIVATA e SIMMETRICA (Cesare, Egizi, ...)
- quella moderna è a CHIAVE PUBBLICA e ASIMMETRICA

la chiave è la K del MODELLO DI RIFERIMENTO

SIMMETRICA vuole dire che la stessa chiave K serve sia per criptare sia per decriptare --> anello debole
ASIMMETRICA vule dire che la chiave NON è la stessa per criptare e decriptare

STREAM CYPHER --> Cesare
'-> considero tutto il testo
BLOCK CYPHER --> DES (Data Encryption Standard)
|-> considero blocchi di testo
'-> esistono diverse versioni
SIMMETRICHE
|-> Electronic Codebook ECB (divido in pezzi, cripto i pezzi e riappiccico, per decriptare al contrario) PARALLELIZZABILE in avanti e indietro
'-> Chipher Block Chaining CBC (XOR prima con blocco NOTO e poi con gli output delle fasi precedenti alla attuale, poi riappiccico, per decrptare al

contrario)
PARALLELIZZABILE solo indietro
ASIMMETRICHE
'->

DATA ENCRYPTION STANDARD

Sviluppato da IBM per il governo Americano --> poi adottato da tutte le istituzioni bancarie negli anni 70 | v le cate bancomat fino a 5 anni fa
E' un BLOCK CYPHER a blocchi da 64b e chiavi da 54b
Ha diverse versioni:
DES
Double DES
Two-Key Triple DES
Three-Key DES

Nel '77 prima dichiarazione di NON sicurezza
Nel '97 rotto per la prima volta in 4 mesi da 4500 macchine che lavoravano assieme verso l'obbiettivo --> ora il tempo si misura in ORE
'-> SOLO LA VERSIONE BASE
Sostituito da AES

ADVANCED ENCRYPTION STANDARD

Usa tre chiavi da 128b, 192b e infine 256b
In BLOCK CYPHER
Usato dagli Stati Uniti per i file secretati

LA CHIAVE PUBBLICA

Teorizzata nel '77 da Deefy e Dellman, che presero il premio Turing
Ognuno ha DUE chiavi: una pubblica e una privata
* sono genrate a coppie
* le cifrate con la privata possono essere decifrate dalla pubblica
* le cifrate con la pubblica possono essere decifrate dalla privata
* SONO UNICHE e nessun'altra chiave può fare il loro lavoro

Ogni utente A vuole comunicare con utente B:
* A rende pubblica al MONDO una delle sue chiavi
* B cripta il messaggio usando la chiave pubblica di A
* A decripta usando la sua chiave privata (l'unica che può decriptare la pubblica)
+ B ovviamnete farà la stessa cosa
+ NESSUNO tranne A e B potrà leggere la conversazione FINCHE' le chiavi private rimarranno tali e DOVRANNO rimanere tali

Shamir, Rivest e Adleman resero possibile la visione di Deefy e Dellman attraverso la FATTORIZZAZIONE DEI NUMERI PRIMI --> RSA (le iniziali)
Esistono anche altri come El Gamal e DSS

La chiave privata può cifrare --> tutti quelli con la pubblica possono decifrare
PROBLEMA DI CONFIDENZIALITA'
Allo stesso tempo però CONFERMO che sono io perché sono il SOLO possessore della chiave privata --> FIRMO DIGITALMENTE
Garantisco la AUTENTICITA' e mi impedisco la NEGAZIONE in campo giuridico
|
v
la NON REPUDIABILITA'

PROBLEMI DELLA CHIAVE PUBBLICA

* MITM può mettersi in mezzo e dare/ricevere la sua/altrui chiave e intercettare le comunicazioni facendo sembrare che tutto sia ok
* 10k volte più lenta di quella a chiave privata (chiavi da 4kb)