
APPUNTI DI SICUREZZA E PRIVATEZZA

CORSO 2023
appunti di Giacomo Castellana

"Measures and controls that ensure confidentiality, integrity and availability of information system assets including hardware, software, firmware and information being processed, stored and communicated"

- National Institute of Standards and Technology

Distribuiti sotto licenza CC-BY-NC-SA 4.0

BASTI SICUREZZA

TIMELINE DEGLI ATTACCHI PIÙ IMPORTANTI

- X 1986 - Hacker tedeschi al soldo del KGB rubano dati all NSA *
- |
- X 1988 - Internet Worm di Morris - Buffer Overflow
- |
- X 1994 - Attacco di Mitnick - TCP Hijacking
- |
- X 1995 - Attacco bancario alla Citibank di Londra da parte di hacker russi
- |
- X 2010 - Stuxnet manda KO e ritarda di anni il programma nucleare iraniano *
- |
- X 2017 - WannaCry sfrutta Eternal Blue (software NSA) per fare un attacco Ransmoware *
- |
- X 2020 - SolarWinds intercetta per mesi le informazioni di svariate organizzazioni e le manda a sconosciuti - Spyware
- |
- V

Da notare come quelli segnati con * sono legati direttamente o per sospetto fondato o per collegamento ai servizi segreti

TERMINI PER INDICARE CHI LAVORA CON LA SICUREZZA INFORMATICA

CRACKER o BLACK HAT --> Hacker malevolo
 GRAY HAT --> Hacker indeciso
 WHITE HAT --> Hacker benevolo
 SCRIPT KIDDIE --> Principianti che usano codice altrui senza conoscerne il funzionamento

I PILASTRI DELLA SICUREZZA INFORMATICA

Le fondamentali sono dette le TRE REGOLE AUREE o AAA dalle loro iniziali inglesi o dal simbolo chimico dell'oro Au:

- * Authentication
- * Authorization
- * Audit

Sono tre concetti fondamentali per la sicurezza in un sistema informatico

Altre tre regole fondamentali e legate alle auree sono:

- * Availability o Disponibilità --> la capacità del sistema di gestire l'accesso alla informazione
- * Confidentiality o Confidenzialità --> la capacità di garantire l'accesso solo agli utenti abilitati
- * Integrity o Integrità --> la capacità di proteggere da modifiche e distruzioni di informazione da parte di utenti non autorizzati

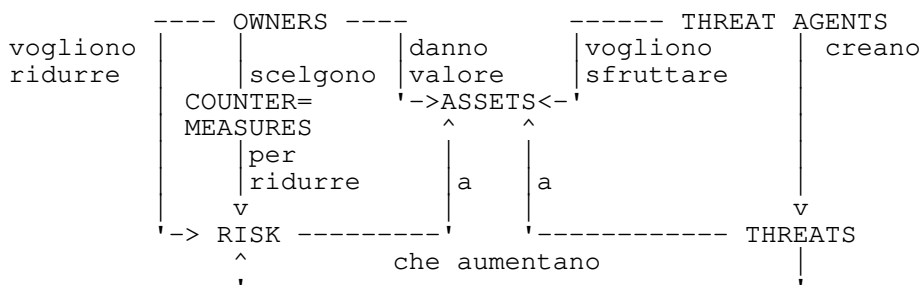
Queste tre regole possono essere meglio specificate a seconda di dove sono applicate:

	Availability	Confidentiality	Integrity
HW	Sabotato	Rubato	/
SW	Cancellato	Copiato	Modificato
Dati	Cancellati	Letti/Analizzati	Modificati
Linee di Comunicazione	Interrotte	Traffico letto	Traffico alterato

ASSET DI UN SISTEMA

- * HW
- * SW
- * Dati
- * Linee di comunicazione

SCHEMA DELLA LOGICA DI SICUREZZA



TERMINI

- * Vulnerabilità --> debolezza di un sistema, sia HW che SW
- * Threat --> minaccia ad un sistema
- * Controllo --> metodi di difesa da Threat
- * Attacco --> sfruttamento delle Vulnerabilità da parte di una Threat
- * Exploit --> Attacco che usa Vulnerabilità
- * Contromisure --> parte del Controllo, mira ridurre il rischio causato dalle Vulnerabilità, ma può causarne di nuove

TIPI DI ATTACCO

PASSIVI --> non intaccano il funzionamento, per questo MOLTO difficili da scoprire
 |
 --> usati per la raccolta di informazioni

ATTIVI --> intaccano il funzionamento del sistema, più facili da scoprire

STEP NEL CONTROLLO

- * delineamento di una Security Policy
- * implementazione della Policy
- * valutazione della Policy e della implementazione da ente esterno
- * test della Policy e della implementazione

POLITICHE DI ACCESSO

Discretionary Access Control (DAC)

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

Mandatory access control (MAC)

- policy dettata dal sistema e decisa per tutti a priori (ambiente militare)
- Role Based Access 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?

ACCESS CONTROL LIST

Divido le persone/utenti secondo il loro ruolo e posizione (se ho 50k utenti ma essi fanno parte di 5 gruppi da 10k -> anziché 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 contenete i metadati dei file

CAPABILITY

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

ACL vs CL

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

AUTENTICAZIONE - la banca controlla chi entra

PARTECIPAZIONE - la banca partecipa

FALSIFICAZIONE - l'elenco è al sicuro

DELEGAZIONE - bisogna aggiungere una 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 LIST

AUTENTICAZIONE - la banca non controlla --|

PARTECIPAZIONE - la banca non partecipa --|

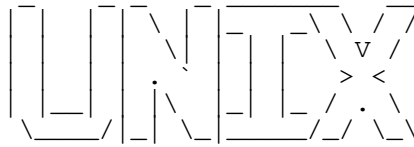
FALSIFICAZIONE - la chiave non è modificabile/copiabile

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 PIÙ



Evoluzione di MULTICS (Multiple Information and Computing System), che a seguito del fallimento della azienda produttrice è stato rinominato, dopo una riscrittura in C, in UNICS (UNIplicated Information and Computing System) nel '74 Per comodità venne chiamato poi UNIX data l'assonanza tra CS e X

UNIX era Open Source e grazie a questo vennero aggiunte svariate modifiche tra cui il MULTITASKING

Nel '78 AT&T comprò la versione 7 del Kernel e la rinominò SYS V rendendolo PROPRIETARIO

L'università di Berkeley creò dal Kernel 6 BSD che rimase Open Source

Dalla fusione di SYS V e BSD nacque LINUX nel '91

L'INTERPRETE DEI COMANDI o SHELL

Detto anche terminale, parla direttamente con il Kernel dell'SO per interpretare ed eseguire i comandi inseriti al suo interno creando durante l'esecuzione processi.

PROCESSO

Detta così l'esecuzione di un programma

Ad ogni processo sono allocate risorse (CPU, memoria, I/O)

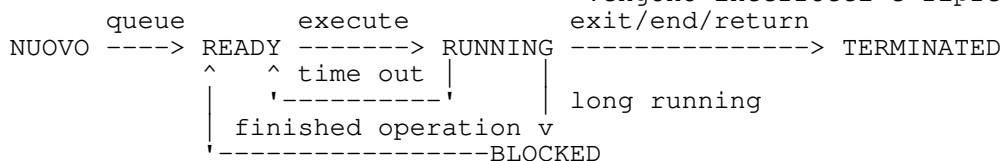
Può essere creato da diverse cose:

- * richiesta dell'utente
- * da un altro processo
- * dal sistema stesso

Quando un processo viene aggiunto esso viene caricato in memoria, gli viene riservato spazio nello stack per le variabili dinamiche e nella memoria statica per le costanti, viene creato un Process Control Block (PCB) e infine viene messo in coda.

Quando termina vengono deallocati tutti gli spazi di memoria, chiusi i file usati e restituito spazio al processo che lo ha chiamato

Vengono gestiti dagli SO in Round Robin --> gestiti a turno ripetuto
 '-> in caso di processi lunghi essi vengono interrotti e ripresi



FORK()

Syscall per generare un processo

Crea un nuovo processo detto FIGLIO identico al creatore detto PADRE in tutto tranne per un valore detto PID. Restituisce -1 se trova un errore, altrimenti restituisce 0 al FIGLIO e il PID del FIGLIO al PADRE

PROCESSO ZOMBIE

Si dice ZOMBIE un processo dimenticato dal padre che si chiude senza prima chiudere il figlio.

Vengono eliminati periodicamente dal SO

PROCESSI IN UNIX

Hanno tutti un PID univoco e possono essere in:

- * Foreground --> interagibili

* Background --> non interagibili senza un eventuale passaggio in Foreground

FILESYSTEM UNIX

Diviso in:

- * Drive --> dischi
- * Directory --> cartelle
- * File --> documenti

Il Drive base si chiama Root ed è indicato con /
/ è anche usato per dividere nel Path (il percorso) Directory superiori da elementi (sia Directory sia File) inferiori

```

/source/dev/....../ciao.txt
  ^       ^       ^       ^
  |       |       |       |
root    First Level Second Level File
Directory Directory

```

Il path può essere ASSOLUTO, parte da Root, o RELATIVO, parte dalla Directory Corrente

L'FS di Unix possiede delle cartelle speciali che contengono programmi e informazioni importanti per il sistema

Si usa il comando cd per cambiare livello di directory e in particolare, oltre al Path desiderato esistono anche questi comandi speciali:

- * / per tornare a Root
- * . usabile per indicare il Path RELATIVO
- * .. per tornare indietro di una cartella
- * ~ per indicare la cartella HOME dell'utente

Esistono altri comandi per il FS come FIND, WHICH, LOCATE, CAT, MORE, HEAD e TAIL

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 contiene groupname:password:GID:list_of_users

SOGGETTI

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

```

:real UID 1001  p1 <-----'
:effective UID 800
                    |
                    | -comando-
                    |
-----> P2 -----' :real UID 1001
                    | :effective UID -800- -->
                    | 462 (esegue il processo con
                    | UID 462)
                    |
                    | <----\
                    | ----->P3 :real UID 1001
                    | :effective UID 462

```

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 ->owner 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 numero di riga

Grep usa anche le espressioni regolari per fare le ricerche
 [:::] dove ::: sono caratteri -> grep cercherà tutte le 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,

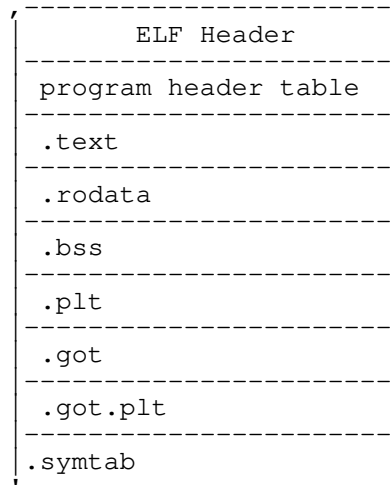
Formato BINARIO

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 successive

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 l'esecuzione

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 #! allora 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 dà controllo
    * se è STATIC-LINKED il kernel lo carica

readelf -a cookie | grep interpreter fa vedere che interpreter 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'SO 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 associato
 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 REAPED

Un processo diventa 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();

PRIVILEGE 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 (finché 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'è una S (vedi sopra)

BASH

Scripting IN Console
Molto potente

ASSEMBLY

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

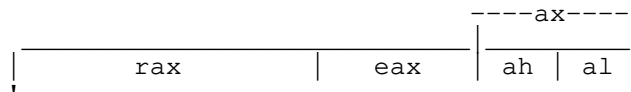
Registri sia a 64b, che 32b, che 16b che 8b a seconda di quali bit voglio

ESEMPIO

```

rax = 64b
eax = 32b
ax  = 16b -> ah+al (higher e lower)
ah/al = 8b

```



dati salvati in LITTLE ENDIAN (al contrario)

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 BISOGNO DI RISERVAMENTO DI SPAZIO IN MEMORIA

ESEMPIO

```

buffer:      resb    64 #byte
wordvar:     resw    1 #word
realarray:   resq    10 #array

                        db      0x55 ->85 numero 0 '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

* Register addressing -> muove i dati da e in
 i registri --> mov rdx,rcx

+ a registro

mov rdx,rcx

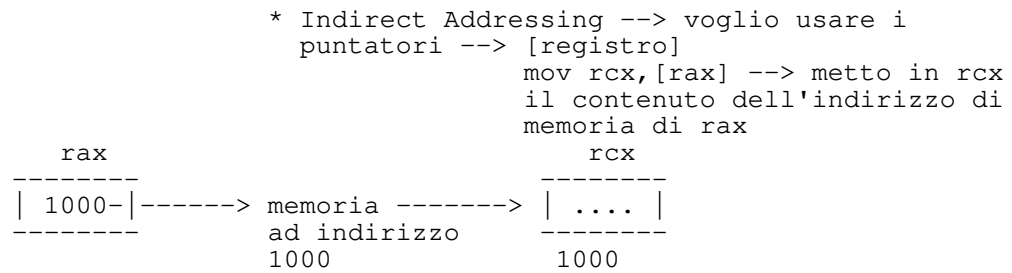
+ a immediato

mov rax,100 --> NON COSI',

è vago, bisogna specificare

mov DWORD PTR rax,100 --> usa

solo i primi 32b



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

```

GIOCARRE 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  #indirizzo array
sumloop:
        mov
        add
        ....

```

LEA Load Effective Address -> carica in 1 l'indirizzo della variabile

```

lea    rcx,buffer |          carico in rcx 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 da specificare
div    dest/src    <-----
idiv   dest,src    #come imul

```

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 stdin)
                    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,rsi #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 del IF)
uso gdb -q nome
la runno finché non segmenta
faccio la info proc map
x/s $rip --> 0x7fffffffdb20:

```

```

"ciao" -> CONTENUTO
della
stringa
COLPEVOLE
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
```

ESERCIZIO

```

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 0 dal programmatore 0 dal compilatore

rispetta naturalmente 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

jj	--> bar
ii	
iiii	
iii	--> foo
ii	
10	
x	--> main

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

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

RET                        --> pop rip
```

PASSARE I PARAMETRI

Fino a 6 argomenti 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 istruzioni 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 iniziale prima che questo venga modificato dalla sotto-routine

A COSA SERVE L'EPILOGO?

serve a annullare il prologo riportando lo stack alla situazione 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

Devo fare l'hijack del flusso di controllo

| FOTO/rethijack.c scopri quale xx e modifica +8 a


```

'-----> è un blocco nello stack di dimensione variabile
           e segreta --> --> serve a obbligare alla
           modifica quel pezzo
canary in inglese <-----'
|'--> il canarino delle miniere
|'--> esempio di canarino: 0x00dba000 <-----'
|
|       se viene trovata una modifica blocca la esecuzione
|--> 00 termina gets()
|--> a0 termina scanf()
|'--> detto TERMINATOR
|
|       v
|       |
|       |--> segna di errore
|       |       di stack
|       |       smashing

```

LATO HARDWARE:
(ma anche SW)

```

Non permettere l'esecuzione dallo Stack
|
|'--> Not eXecute Bit in AMD
|'--> Execute Disabled Bit in Intel
|--> si contravviene leggermente il principio
      di Von Neumann e si decide che lo stack
      NON può contenere codice
|
|--> -z execstack rimuove l'implementazione SW
|--> su Windows implementato con DEP (Data Execution Prevention) su XP
      e server 2003 --> evita che il codice venga eseguito dallo stack
      e heap
|'-----> sia SW che HW

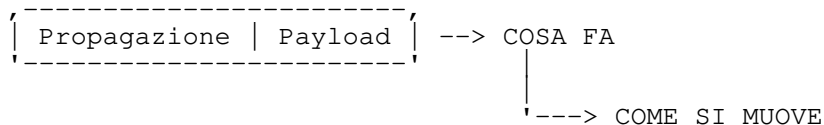
```

MALWARE

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

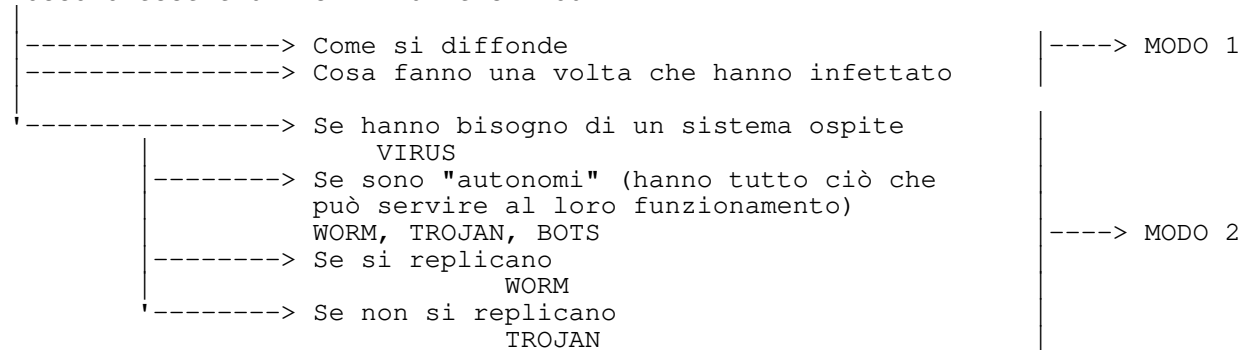
Diversi tipi --> Worm, Trojan, Backdoor, Keylogger, Logic Bomb, ...

COMPOSIZIONE DI UN MALWARE



CLASSIFICAZIONE

Possono essere divisi in diversi modi



PROLIFERAZIONE

Infezione di altro software
Sfruttamento di un exploit
Social Engineering 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 --> generano il codice malevolo a partire da poche informazioni

```

  ----> Zeus e Angler
  ----> Comprabili da soggetti poco raccomandabili
  
```

CHI LO FA

* Chi ha motivazioni politiche	--> dittatori, governi
* Criminali solitari	--> 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 PERSISTENT 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

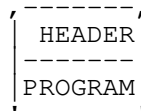
VIRUS

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 Privilege
|                  Escalation

```



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

FASI

```

IDLE/DORMIENTE --> non fa nulla finché non c'è un trigger
                --> non sempre presente
ATTIVAZIONE    --> il trigger avviene (se presente)
PROPAGAZIONE   --> si riproduce (non sempre)
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 PIÙ --> sono stati
                    sviluppati
                    VIRUS CIFRATI
                    |
                    v
prima c'è il virus criptato POI la
routine di decriptazione e criptazione
                    |
                    v
sgamabile attraverso 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 osservando cosa fa --> se fa cose sospette lo
        | flaggo
        | '--> ANALISI DINAMICA

```

CLASSIFICAZIONE

DIPENDENTE DAL BERSAGLIO

```

        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

```

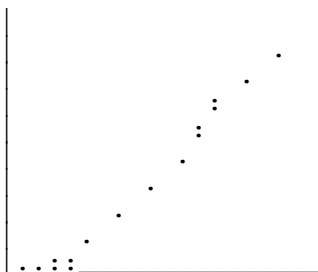
In generale un Worm attacca una macchina (che ha un certo IP) e da qui si propaga in diversi modi

```

| --> 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

RANSOMWARE

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 individuata una vulnerabilità su un Browser, si fa scaricare (al momento della connessione ad un server) un documento che si auto-esegue 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 ENGINEERING

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

COSA PUÒ 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
--> controlla contatti
--> tiene sotto controllo le immagini
```

----> e le invia a chi di dovere

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
Simile al Social Engineering

BACKDOOR

Punto di ingresso nascosto nella macchina ad uso futuro creato DURANTE la creazione
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 permessi 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 MODE --> 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 divide 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
    |--> Analizzando in automatico con Machine Learning, AI e
    |     Threat Intelligence
    |--> analisi

```

```

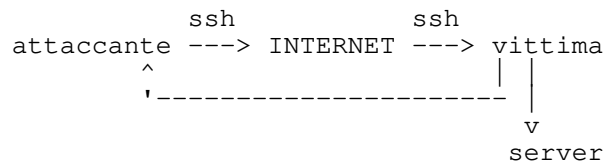
    detti Next Generation Antivirus <-----'

```

ABRA WORM

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

SCHEMA



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 "abracadabra"
    |--> 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 docker (VM)

```

|--> vedi guida qui: https://git.laser.di.unimi.it/teozio/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 RANSOMWARE con libreria crypto? --> TBD

```

NETWORK SECURITY

Nuove tecniche 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 calcolatori 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 manufacturer)

--> 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 telefonico
            '-> 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
    APPLICAZIONE
    PRESENTAZIONE
    SESSIONE
    TRASPORTO
    NETWORK
    DATA LINK
    FISICO

```

ARP e ARC

```

ARP request vuole mandare una cosa --> chiede indirizzo, chi lo ha
                                        |
                                        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'indirizzo 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 servizio 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, anziché 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

quando passa al livello 2 (internet) PRIMA si
 controlla se il destinatario è sulla LAN
 |->usa una tabella

|-> ARP CACHE che relaziona IP a MAC

| IP addr | MAC addr |

--> se lo trova e appiccica al pacchetto IP la sua parte (MAC src e MAC dst) e lo manda sulla LAN a tutti

| |MACsrc|MACdst|IPsrc|IPdst| IP | data |

|-> il destinatario lo riceve e lo legge

|-> se NON lo trova, prima di mandare al router, fa una ARP request a TUTTI sulla rete --> "chi ha questo IP?"

|MACsrc|BROADCAST|IPdest?|

chi lo ha risponde con questo pacchetto

|MACsrc| |IPdest|

-----'

MITM: MAN IN THE MIDDLE

L'attaccante si mette in mezzo tra due host e intercetta le informazioni scambiate da essi, arrivando anche a modificare 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)

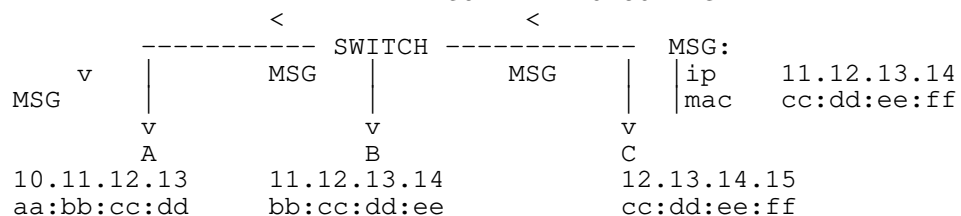
'-> evita di dover fare continue richieste ARP

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)

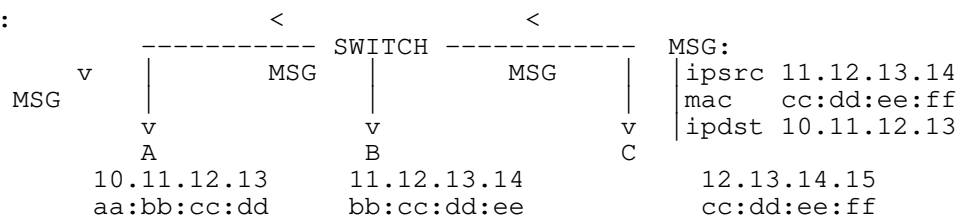
E' HALF perchè intercettiamo solo ciò che esce da A verso B, non anche da B ad A

FULL MITM

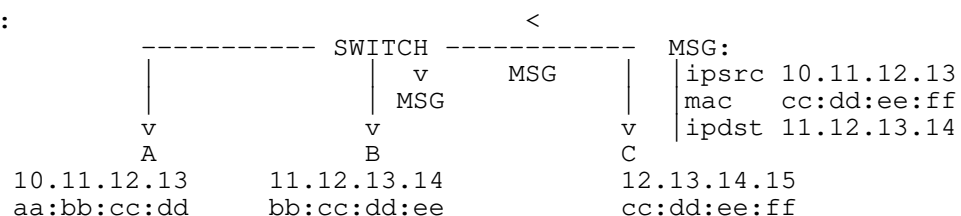
Intercetta sia da A a B sia da B a A

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 risponde 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 DISPONIBILITÀ 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/sistema 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 possono 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

Il SYN FLOOD di Mitnik

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
HOST  --->----->                --- X e Y sono generati a caso
      ' _ '
                                <----- ' _ ' 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

VULNERABILITÀ

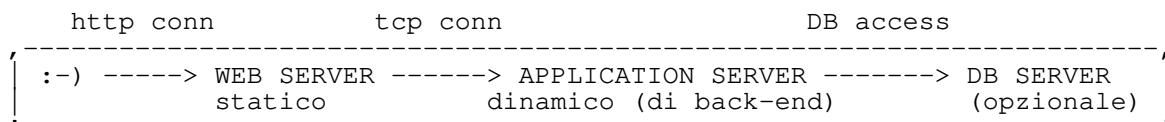
Possono esistere metodi di input NON controllati, vulnerabili alla Injection
--> Javascript ad esempio --> xss payload
Ci si può difendere facendo filtraggio dell'input --> ci sono librerie, non fare a mano

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 leggere file con pg_read_file

Payload SQL Injection
SQLmap automatizza

Esistono, come per html, dei metodi 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)
```

HOST PROTECTION SYSTEM

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

--> Authentication --> 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

Enforcement 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 coincidono con i processi

Objects --> entità passive --> subiscono 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

v

è in due fasi

1) autocertificazione dell'utente -----,

2) controllo della affermazione dell'utente

v

1 è diviso in diverse tipologie a seconda delle organizzazioni
(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 ricondividere
 * 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

_	--> dammi la password --> :-)	RICHIESTA
--		
computer	utente	
_	<-- ciaoCIAO <-- :-)	RISPOSTA
--	password in chiaro	
computer	utente	
..	<-- lmekCALM <-- _	CRITTOGRAFAZIONE
..	criptazione	--

file criptato	computer	

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

WHAT YOU KNOW - VULNERABILITÀ

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 Engineering --> conoscere la persona per prevedere la password
 '--> appassionato 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 crittazione è noto (md5, ...)
 '-> la difficoltà è ESPONENZIALE (3h per 6 caratteri, 2 anni per 8 caratteri, ...)
 ,
 '-> di norma si restringe il campo su parole 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 hash-ate

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 Failure (se ho un taglio sul dito non posso usare il lettore)
 - * Può fare errori di lettura (sensore sporco)
 - * Lettura lenta (deve leggere 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'

MULTI-FACTOR AUTHENTICATION

SINGLE SIGN ON

FEDERATED IDENTITY MANAGEMENT

OPEN ID

ACCESS CONTROL

- * Discretionary Access Control - DAC
- * Role Based Access Control - RBAC
- * Mandatory Access Control - MAC

MAC ACCESS CONTROL

```

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

```

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:

$$+ 0 \rightarrow i1 + potente$$
$$+ 1$$
$$+ 2$$
$$+ 3$$

sui sistemi più comuni

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 Privilege
  Level --> specifica la etichetta --> dettato da SO
```

- + esiste anche un CPL Current Privilege 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.

- * $CPL \leq DPL$ 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 Privilege 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
 $(MAX(DPL_B, CPL_A)) \leq DPL_D$? 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 Kernel 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 interrotto, 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

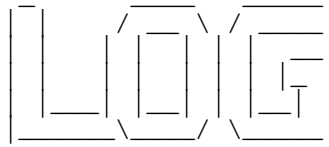
| '--> Interrupt Descriptor Table il rischio di venire
 IDT disturbato dal software
 utente, che è fermo

'--> Un attacco può modificare 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

Esaminazione 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 PC normale 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 anziché 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 vulnerabilità

'--> chiamati LMS (Log Management Systems) o più modernamente SIEM (SECURITY, INFORMATION, EVENT MANAGERS)

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

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



COSA E'

La scienza della occultazione reversibile delle informazioni attraverso delle trasformazioni MANTENENDO l'informazione originale

IL SUO CONTRARIO

CRIPTO ANALISI --> scienza che verte sulla reversione di un oggetto criptato

SI BASA SU

Confidenzialità
Irrepudiabilità
Integrità

MODELLO DI RIFERIMENTO

```

      K
      |
      v
:-) --> bla bla --> Funz Critt --> wsysytsyt --> :-)
      ^
      |
    >:-(
  
```

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 vuole 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

'-> Cipher Block Chaining CBC (XOR prima con blocco

NOTO e poi con gli output delle fasi precedenti alla
attuale, poi riappiccico, per decriptare 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 carte 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 segretati

LA CHIAVE PUBBLICA

Teorizzata nel '77 da Diffie e Hellman, che presero il premio Turing
Ognuno ha DUE chiavi: una pubblica e una privata
* sono generate 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 decrypta usando la sua chiave privata (l'unica che può decriptare
la pubblica)
+ B ovviamente farà la stessa cosa
+ NESSUNO tranne A e B potrà leggere la conversazione FINCHÉ le chiavi
private rimarranno tali e DOVRANNO rimanere tali

Shamir, Rivest e Adleman resero possibile la visione di Diffie e Hellman
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 CONFIDENZIALITÀ
Allo stesso tempo però CONFERMO che sono io perché sono il SOLO possessore della
chiave privata --> FIRMO DIGITALMENTE
Garantisco la AUTENTICITÀ e mi impedisco la NEGAZIONE in campo giuridico

|
v
la NON RIPUDIABILITÀ

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)

PROBLEMI GENERALI DELLA CRIPTAZIONE

Il messaggio può essere modificato da vari fattori volontari o meno durante la trasmissione che corrompe i dati e rende diverso il messaggio decriptato

ONE-WAY HASH FUNCTIONS

Un Hash è una funzione che mappa dati ARBITRARI (stringa, file, ...) ad un valore di dimensione fissata (128b-512b) chiamata FINGERPRINT (indipendentemente dalla lunghezza originale dell'oggetto)
Le funzioni di Hash perfette sono a UNA VIA --> cioè che sono MOLTO difficili da invertire

|
v
praticamente impossibile

PROPRIETÀ

- * ONE WAY (vedi su)
- * COLLISION RESISTANT --> difficile che esistano due hash uguali da due file m1 e m2 diversi
 $\text{hash}(m1) = \text{hash}(m2)$ MOLTO DIFFICILE
 quasi un secolo <-'
 -> questo le rende suriettive <- tanto in poco
 -> utile a capire se un file è corrotto durante il download
 -> le password su etc/shadow sono salvate in questa maniera
 |
 v
 * utente inserisce la password
 * computer la hasha
 \$ confronta con la hash originale
 + se sono uguali entra
 + se sono diverse non entra

APPLICAZIONI DELL'HASH ALLA CHIAVE PUBBLICA

Può velocizzare il processo di autenticazione della firma digitale tramite chiave privata attraverso l'hashing della cosa da firmare prima di firmare

oggetto O --> $\text{hash}(O)$ --> $K_{\text{priv}}(\text{hash}(O))$

Per validare si riceve la firma e l'oggetto O. Si usa la chiave pubblica per decriptare la firma e ottenere la hash dell'oggetto, dopo di che si hasha O e si confrontano l'hash(O) NOSTRO con l'hash(O) FIRMA --> se combaciano si conferma l'identità

HASH PIÙ COMUNI

MD5 --> la sua collision resistance rotta nel 2005
 SHA1 --> la collision resistance è considerata compromessa (è possibile in 2^{80} tentativi)
 SHA2 --> ora come ora la collision resistance è considerata sicura

AGGIUNGERE PIÙ INTEGRITÀ

KEYED-HASH MESSAGE AUTHENTICATION CODE

Si usa la funzione di hash e una chiave K

```

1:      hash(| K xor ipad | messaggio |)
          |
          v
2:      hash(| k xor opad |          |) -> HMAC(K,M)

Si riceve M e HMAC(K,M), si estrae da 2 1 e si riefettua 1
se 1_rifatto e 1_originale sono uguali tutto ok -----

```

COME SI PUÒ CONFERMARE LA CHIAVE PUBBLICA

Due metodi

- * Quando si presenta la propria chiave pubblica si presenta anche un documento che attesta la paternità della stessa AD UN ENTITÀ NOTA e AUTOREVOLE
 '--> come la CdI
- * Stessa cosa di prima ma ci si fida del giudizio degli altri utenti della rete

Questi due metodi hanno generato tre certificati

'-> PKI - Autorità
 '-> X509 - Autorità
 '-> PGP - Libero

PKI e X509

Si basano entrambi ad una Certification Authority che, alla presentazione di una chiave pubblica esegue dei controlli e certifica o meno la paternità della chiave attraverso un CERTIFICATO DIGITALE

'-> firmati con la chiave privata DELLA Cert. Auth.
 '-> si valida con la chiave pubblica DELLA Cert. Auth. --> CIRCOLO VIZIOSO
 '-> come mi fido?

↓
v

la Cert. Auth. rilascia in formato
 cartaceo le hash
 '-> quelli dello SPID sono su Gazzetta
 Ufficiale

Al rilascio di una chiave essa viene infilata in una banca dati chiamata
 PUBLIC KEY INFRASTRUCTURE

--> HTTPS e le connessioni protette

* chiede alla prima connessione AL SERVER il certificato
 * il server lo manda AL BROWSER
 * controlla a CHIAVE PUBBLICA delle CA installate sul browser

↓
v

ogni browser ne ha
 diverse
 '-> alcuni permettono di
 aggiungere altri
 certificati

--> usato sulle Carte di Credito a CHIP

* esse possiedono un chip con una coppia di chiavi, una privata l'altra pubblica

ALLA AUTENTICAZIONE

+ la CARTA invia il certificato a chiave PUBBLICA
 + il TERMINALE contiene il certificato con la firma e lo usa per
 verificare la bontà di quello inviato dalla carta
 Ora deve controllare la PRIVATA della carta attraverso l'invio
 di un numero casuale R che la carta deve firmare
 + la CARTA firma il numero R con la PRIVATA e lo invia
 + il TERMINALE verifica R_firmato con la chiave PUBBLICA della
 carta
 # se è vero TUTTO OK
 # altrimenti NULLA

ALLA TRANSAZIONE

+ il TERMINALE manda alla CARTA i dati della transazione
 + la CARTA firma con la PRIVATA e manda al TERMINALE
 + il TERMINALE conferma attraverso la PUBBLICA e manda alla
 banca

Testi di approfondimento: - Applied Cryptography

APPLICAZIONI DELLA CRITTOGRAFIA ALLA NETWORK SECURITY

RFC --> Request For Comments

|-> descrizione di un protocollo da implementare
 '-> scritti da gruppi di grandi guru della sicurezza (gente di Cisco,...)

PROTOCOLLI

DIFFLE-HELLMAN KEY EXCHANGE (quelli di chiave pubblica-privata)

Algoritmo elaborato per lo scambio di un segreto su canale IN CHIARO
 Basato su un problema matematico --> logaritmo discreto su campi finiti

↓
v

un logaritmo dato $b^Y=c$ permette di ottenere Y

↓
v

poniamo che ci siano n, x, b tali che $b^Y=x \pmod n$

quella Y è il Problema del Logaritmo Discreto
-> NON CI SONO ANCORA ALGORITMI EFFICIENTI
-> possibile solo con primi e molto lunga
-> praticamente One Way

STEP:

- * A e B devono condividere un segreto e decidono un primo p e un alfa tra 1 e p
- * entrambi generano due numeri casuali r_A e r_B che mantengono segreti
- * A calcola $n_A = (\text{alfa}^{r_A}) \pmod p$
- * B calcola $n_B = (\text{alfa}^{r_B}) \pmod p$
- * si scambiano i loro risultati
- * A calcola $n_B^{r_A} = ((\text{alfa}^{r_B})^{r_A}) \pmod p = \text{alfa}^{(r_B \cdot r_A)} \pmod p$
- * B calcola $n_A^{r_B} = ((\text{alfa}^{r_A})^{r_B}) \pmod p = \text{alfa}^{(r_A \cdot r_B)} \pmod p$
- > A e B possono quindi calcolare uno stesso numero

ESEMPIO

A e B scelgono $p=23$ e $\text{alfa}=11$
A sceglie $r_A=6$
B sceglie $r_B=5$
A calcola $11^6 \pmod{23} = 9$
B calcola $11^5 \pmod{23} = 5$
A riceve da B 5
B riceve da A 9
A calcola $5^6 \pmod{23} = 8$
B calcola $9^5 \pmod{23} = 8$

CRYPTO E CYBER SECURITY

CONFIDENTIALITY

Crittografia

INTEGRITY

Attacco un HMAC all'oggetto

AUTHENTICITY

Firma digitale

PROBLEMI

Politici:

Trattato di Wassenaar --> i paesi si sono impegnati post WW2 a diverse cose tra cui GLI ALGORITMI DI CRITTOGRAFIA

↓
Ha causato un problema alla RSA perché NON si poteva esportare fuori dagli USA normalmente Solo "depotenziata" (in Europa a 128b in USA a 1028b)

Quello di PGP è stato arrestato perché ha infranto il trattato

'-> ha poi raggirato perché solo il BINARIO è reato, non il sorgente SU LIBRO

↓
Sono stati liberalizzati per le banche

Tecnici:

Performance: se lo voglio rapido non è sicuro

Sicurezza: se lo voglio sicuro non è rapido

Implementazione: quando lo faccio?

IMPLEMENTAZIONE

Posso farlo nel programma prima di mandare avanti i dati ai layer sotto
Posso farlo sul sistema nei livelli inferiori a quello applicazione

A seconda di dove lo faccio ha anche diversi gradi di granularità e di performance --> quanto la voglio SICURA?

AL LIVELLO NETWORK

In blocco, cifro tutto ciò che passa dal livello IP, indipendentemente da cosa stia arrivando

Non proteggerò da IP in su

Costa in termini di tempo di CPU, la crittografia è pesante

E' leggero per il programmatore perché non ci deve pensare --> utile se non si è bravi nel programmare

Le VPN sono l'esempio

```
| IP header | AdgswugekghxixmDEQIUIEIDHKNwaiudh |
maschero un pezzo di IP Header
```

AL LIVELLO TCP

Meno leggero rispetto al Network per il programmatore

Garantisco che il dato è protetto fino al livello di trasporto

Gmail è l'esempio

```
| IP header | TCP header | ahjdahdkjJahdHShdkjw |
maschero un pezzo di TCP Header
```

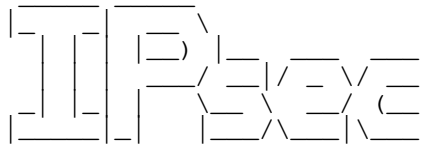
AL LIVELLO APPLICAZIONE

Pesante per il programmatore ma possibilmente più leggero per la CPU, che cripta solo ciò che necessita crittografia

Garantisco che il dato sia protetto FINO all'utente

Whatsapp è l'esempio di questa cifratura detta End2End

```
| IP header | TCP header | EAUDWUDiuwqhsjHJWA |
maschero solo il Payload
```



Inizialmente era un sottomodulo di IPv6 (vedi nella sezione della rete)
Poi è stato estratto e aggiunto a IPv4 per necessità di aggiungere sicurezza

Fa le determinate cose (solo una, solo l'altra o entrambe):

- * cripta il traffico al livello IP
- * decrypta il traffico al livello IP
- > mira a proteggere dal packet sniffing

Nella versione a blocchi cifrati usa o AES-CBC o Triple DES-CBC

Nella versione a hash usa

Offre i seguenti servizi:

- * Data Origin Authentication
- * Confidentiality
- * Connectionless and partial sequence integrity
 - + Connectionless: certifica la non modifica del pacchetto
 - + Partial sequence integrity per evitare i reply attack
- * Limited Traffic Flow Confidentiality
 - + non permette di sapere da fuori chi parla
- > tutti TRASPARENTI ai livelli successivi --> non bisogna preoccuparsene

I protocolli sono:

- * Authentication Header --> autenticazione di tutto il pacchetto
- * Encapsulated Security Payload --> autenticazione del solo payload
- > più usato è ESP, AH è opzionale, prima invece era obbligatorio e la soluzione "default" --> from MUST to MAY

Ha due metodi di incapsulamento: trasporto e tunnel

pacchetto base | IP header | TCP header | dati |

pacchetto in
modalità | IP header | IPsec header | TCP header | dati |
trasporto

pacchetto in
modalità | IPsec header | pacchetto IP |
tunnel

La modalità tunnel serve come scalino di metà tra IPv4 e IPv6
| -> reincapsula un pacchetto IPv6 in un pacchetto IPv4 così che
| eventuali strutture di rete che non supportano IPv6 possano
| comunque inviare i pacchetti
| -> usato dalle VPN

AH

,---> parametro per identificare la SA del mandante

SPI e Integrity Check

| -> si fa un doppio SHA1 del pacchetto e lo si infila nel campo
| (il capo IC è di conseguenza vuoto prima del calcolo)
| -> parente del HMAC

AH IN TRANSPORT MODE:

Autentica l'IP payload e alcune parti del IP header e alcune
parti della estensione IPv6

AH IN TUNNEL MODE:

Autentica l'intero pacchetto IP più alcune porzioni dell'header
IP

ESP

Aggiunge anche lui un SPI e una specie di HMAC

ESP IN TRANSPORT MODE:

Cripta il payload IP e ogni estensione di header IPv6

ESP IN TUNNEL MODE:

Cripta il pacchetto IP interno

ESP CON AH

Fa sia la parte dell'AH sia quella dell'ESP

ESP CON AH IN TRANSPORT MODE:

Cripta il payload IP e estensione di header IPv6 mentre autentica solo l'IP payload ma non l'IP header

ESP CON AH IN TUNNEL MODE:

Cripta il pacchetto IP interno e lo autentica ma non autentica il pacchetto IP esterno

SECURITY POLICY e SECURITY ASSOCIATION

Parametri che fanno decidere cosa fare su:

- * pacchetti (buttarlo, non fare nulla, a che livello criptare, da che porta farli partire,...NOME DELLA ASSOCIAZIONE DI SICUREZZA)-----,
- * che modalità di IPsec usare (tunnel o trasporto)

Deve essere specificata PER OGNI traffico di dati --> tenuti in un DATABASE

Esiste poi un ulteriore database chiamato SECURITY ASSOCIATION DATABASE, che specifica a basso livello COME implementare OGNI policy (come criptare, che algoritmo, che chiave usare, ...)

Inizialmente erano da riempire a mano, poi nel 2005 è arrivato il protocollo IKE che funziona senza intervento umano per riempire i campi automaticamente facendo comunicare le macchine

IN GENERALE

Chi manda:

```

livello IP --> consulta SPD --> se non trovo --> butto
|
v
trovo
/ \
|  \ applico IPsec
|   \ non esiste SA
v     \
consulta SAD -----> IK
| esiste <-----
v SA crea SA
| applica IPsec al pacchetto
| inoltra il
v v pacchetto
rete

```

Chi riceve:

```

rete
| pacchetto IPsec
v
Controlla SAD ----, non è in SA
| -----> scarta il
v -----> pacchetto
Applica SA -----' non soddisfa SPD
| soddisfa SPD
v
Livello IP
|
v
Livelli superiori

```

I BENEFICI DI IPSEC

Fornisce un buon livello di sicurezza per tutte le applicazioni restando però trasparente ai livelli superiori e agli utenti.
Può essere fornito ad utenti specifici ed è pronto a modifiche di criptazione con algoritmi migliori in futuro

LE CONTROINDICAZIONI DI IPSEC

Aggiunge un peso di calcolo per ogni pacchetto rendendo complesso l'invio contemporaneo di un alto numero di pacchetti
Non permette l'uso da chi non l'ha
L'host deve avere una SA, che non è ottimale per connessioni di breve durata

TRANSPORT LEVEL SECURITY

Inventato da NETSCAPE nel 1994 come SSL (Secure Socket Layer), venne poi reso standard con il nome TLS nel 1999. La sua ultima versione è del 2018
 Mira a creare una connessione protetta tra due parti PORTA A PORTA --> TCP
 Predefinito nella maggior parte dei browser e servizi web
 HTTPS usa TLS anziché solo il TCP come HTTP

I PROTOCOLLI

- * Autenticazione Server
- * Autenticazione Client
 - + opzionale
- * Confidenzialità dei dati (connessioni criptate)
 - + utili a evitare intercettazioni
- * Integrità dei dati
 - + protegge da modifiche
- * Generazione e distribuzione di chiavi di sessione
 - + all'interno del protocollo
- * Negoziazione dei parametri di sicurezza
- * Compressione e decompressione

NON fornisce la non-repudiabilità

GLI ALGORITMI

X509 per l'autenticazione server/client

- * RSA
- * DH
- * DSS
- * Fortezza

Criptazione:

- * RC4 (40-128)
- * DES (40-128)
- * TripleDES
- * AES

Hashing:

- * MD5
- * SHA1

DOVE SI PONE NEL PACCHETTO?

Nel pacchetto si pone appena dopo TCP

COME SI DIVIDE?

Handshake --> i peer si accordano su cosa cifrare e come
 Record --> "macina" i dati nel trasferimento

HANDSHAKE

```

C --> Client HELLO, cipher suite proposal      --> S
C <-- Server HELLO, certificate, server key exc <-- S
      cipher suite ok/not ok - proposal
C --> Client key exc, cipher spec, cipher      --> S
      suite ok/not ok - proposal

      ...
C <-- Cipher Spec, Finished                    <-- S

```

Cipher Suite C chiede cosa usare nella connessione (criptazione, chiave)
 " " S accetto o meno (nel caso propone di nuovo)
 Calcolano il Master Secret assieme e poi ci fanno calcoli con SHA, i
 numeri casuali e altri dati per calcolare la chiave segreta
 Dalla chiave segreta se ne tirano fuori 6

```

      -> Client write MAC
      -> Server write MAC
      -> Client write
      -> Server write
      -> Client write Initialization Vector
      -> Server write Initialization Vector

```

Può esserci un MITM --> per evitare oltre alla chiave pubblica si manda
 anche il certificato X509 --> anche il server può chiederlo all'utente

RECORD

Prende un messaggio d'applicazione e ci lavora:

- * lo spacca in blocchi
- * OPZIONALMENTE comprime i blocchi
- * calcola un MAC per i blocchi
- * Cripta i blocchi
- * aggiunge un header
- * trasmette

BUG TLS

BEAST permetteva nel 2011 di leggere i pacchetti non criptati tramite un bug
 del Cipher Blocking Chain
 HEARTBLEED permetteva nel 2014 di rubare le chiavi private tramite un bug di
 OpenSSL

DEBOLEZZA TLS

Non è più trasparente per le applicazioni
 Permette a chi gestisce i servizi finali di leggere i dati

END TO END

Criptazione dal livello applicazione (eventualmente rafforzata da passaggi extra) per rendere perfettamente "sicuro" il passaggio dei dati attraverso i sistemi di mezzo senza interferenze o spionaggio

Whatsapp, Signal, Telegram lo usano (i primi due usano Signal, l'altro MTProto)

MTProto

Creata apposta per Telegram

Suppone a priori la INSICUREZZA di tutti i livelli sottostanti

Prevede due protocolli di criptazione

- * Cloud Chats --> memorizzati in chiaro
- * Secret Chats --> memorizzati cifrati

TELEGRAM CLOUD FUNZIONA COSI':

- * S e A/B determinano con DH una chiave
A --> KdhA --> S <-- KdhB <-- B
- * quando A deve parlare con B, A manda cifrato a S, S lo decripta, S lo cripta di nuovo, S lo manda a B --> S li vede IN CHIARO
- > permette di recuperare istantaneamente i messaggi siccome sono sui server

TELEGRAM SECRET funziona così:

- * A e B, attraverso S e con DH, decidono le chiavi --> S può fare MITM
- _i proprietari_ dicono che è evitabile facendo il <--'
- controllo manuale OUT OF BAND tra i due

Più di 100 messaggi recuperabili con una chiave

SIGNAL

Usa algoritmi complessi (sempre DH) come

- * ECDH - Curve25519 --> basato sulla curva ellittica di Edwards 25519

Signal funziona con server in 4 passaggi:

- * Registrazione --> si salva un plico di roba dell'utente in un database
- * Set-Up di sessione --> quando A deve comunicare con B, A accede alle info di B su S e fa dei conti con esse (3 ECDH e 1 KDF) --> ad ogni ..
- * Comunicazione simmetrica --> A mette il messaggio cifrato su S e B lo prende e fa gli stessi calcoli di Set Up
- * Comunicazione asimmetrica

Meno di 100 messaggi ripristinabili con una chiave

S NON ha nessun mio messaggio, solo il mio dispositivo

WEB OF
TRUST

PGP

Pretty Good Privacy

Inventato da Zimmerman, mira a sorpassare le Cert. Auth. basando le comunicazioni sul fidarsi del prossimo

Rende possibile decidere un livello di fiducia tra gli utenti --> Web of Trust

JOHN THE RIPPER

Password cracker

Prende una lista di hash/uno solo e attraverso vari metodi tenta di tornare al plaintext

! --> sia hash normali sia con salt sia file etc/shadow

DEFENSIVE CYBERSIC

STRATEGIA DI DIFESA

Da adottare per prevenire e ridurre l'impatto di un attacco

Dividasi in 5 fasi:

- 1 Identify - Identificazione delle falle - CRP
- 2 Protect - Protezione dalle falle - CRP
- 3 Detect - Riconoscimento dell'attacco - DCO
- 4 Respond - Risposta all'attacco - DCO
- 5 Recover - Se tutto va male, recuperare il sistema al momento precedente all'attacco - DCO

Cyber Resilience Programme --> 1 e 2 --> prevenzione

Defensive Cyber Operations --> 3, 4 e 5 --> difesa attiva

STRUMENTI BASE DI CYBERSICUREZZA

Ne esistono di diversi:

- * Crittografia
- * Protocolli di crittazione
- * Antivirus
- * EDR (Endpoint Data Recovery)
- * ASLR, Stackguard, ...

Possono non bastare in alcuni casi

FIREWALL

Oggetti sia SW sia HW che analizzano il traffico di rete e decidono caso per caso cosa fare con il traffico che gli passa attraverso

Sono programmati con regole specifiche che determinano il suo comportamento

- * controlla tutto
- * controlla solo i pacchetti che vengono da X
- * se il pacchetto è sospetto avvisami tramite LOG

```

internet      --> | FIREWALL | --> rete interna
                  xx>          -->
                  -->          -->

```

Spesso usati in aziende e organizzazioni

Una volta erano scavalcabili tramite la rete GSM

Possono avvisare di eventuale traffico malevolo o supposto tale

Usato anche come frontiera per dividere sottoreti di una stessa organizzazione

TIPI DI FIREWALL

Ne esistono di diversi tipi:

- * Packet Filter --> guarda solo l'header, non il payload
 '-> "scarta tutti i pacchetti di X.Y.Z.K"
 "fidati di tutti i pacchetti U.O.P.T"
- * Stateful Inspection --> guarda anche il payload
- * proxy Firewall --> si mettono in mezzo alla connessione

- * Application Gateway --> applica meccanismi di sicurezza a specifiche applicazioni come TCP, Telnet

PACKET FILTER

Fa passare o no un pacchetto da una rete ad un'altra, non sulla stessa

Analizza i seguenti campi:

- * IPsrc/dest
- * Protocollo di trasporto
- * TCP/UDP src/dest
- * tipo di messaggio ICMP
- * opzioni di pacchetto

Fa le seguenti cose:

- * fa passare il pacchetto
- * ferma il pacchetto (comunicandolo al mittente o no)
- * modifica il pacchetto
- * logga le informazioni sul pacchetto

E' pesante controllare ogni pacchetto, soprattutto se la connessione è già sicura

STATEFUL FILTERING

Evoluzione del packet filter che evita il controllo di ogni pacchetto ma solo quelli delle nuove connessioni

ESEMPIO DI FIREWALL

ABILITARE LE CONNESSIONI SSH DA ESTERNI A INTERNI

Due regole

- * Inbound
- * Outbound

Se Inbound arriva da >1023 e vuole arrivare a 22

Se Outbound va da 22 a >1023

Se è TCP

--> è SSH

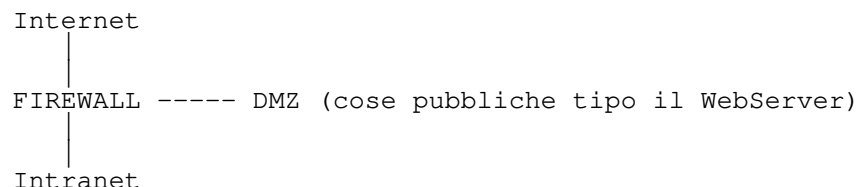
Genera la tabella di regole seguente:

Regole	Dir	Src Addr	Src Prot	Dst Addr	Dst Prot	Prot	Ack?	Action
SSH-1	In	Ext	>1023	In	22	TCP	any	Allow
SSH-2	Out	In	22	Ext	>1023	TCP	yes	Allow

REGOLE DI DEFAULT

- * Egress filtering
- * Ingress filtering
- * Deny default

CONFIGURAZIONE DI BASE



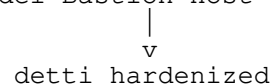
Internet può accedere a DMZ --> accede solo alle cose pubbliche (sito web)

Intranet può accedere a DMZ e Internet --> accede a tutto perché autorizzato

DMZ può accedere a Internet --> così non fa da ponte tra internet e intranet

DMZ

La più vulnerabile agli attacchi --> difesa da dei Bastion Host



|
v

rinforzati grazie ad un kernel
modificato tramite la rimozione di ogni
cosa superflua per il suo funzionamento

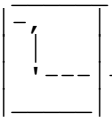
PROXY FIREWALL

Dividono in due la connessione e vi si mettono in mezzo

Di due tipi:

- * livello applicazione --> Web Application Firewall o WAF per HTTP
- * livello trasporto

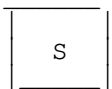
Trasporto --> utile per le trasmissioni cifrate

fuori --->  --> dentro

Applicazione --> utile per analizzare il payload per trovare eventuali
minacce come buffer overflow (troppi x90 di fila)
-> pesante dal punto di vista computazionale
'-> necessità di un firewall PER OGNI applicazione (HTTP,
FTP, ...)

AIR GAP

Un server riceve tutto il traffico e un demone analizza il traffico e decide il
da farsi e manda VERSO, non A, la rete locale ciò che va bene

fuori <-->  <--> dentro

Il server è IN MEZZO dividendo la connessione

Usato probabilmente da NSA e servizi segreti

NEXT GENERATION FIREWALL

Un firewall che non solo controlla la porta e il pacchetto ma analizza, indaga e
previene le intrusioni in modo autonomo usando conoscenze esterne al firewall

Sono anche capaci di riconoscere e bloccare il malware (feature di antivirus)
direttamente dal payload

IDS

Intrusion Detection Systems --> sistemi che rilevano le intrusioni IN TEMPO
REALE e in maniera autonoma tramite sistemi HW e
SW
-> sono PASSIVI --> avvisano e basta
'-> stanno iniziando ad essere ATTIVI --> difendono
e magari attaccano a loro volta

Divisi in due tipi

- * Host Based --> su macchine individuali (computer, server, ...) tramite
programmi-sensori (SW) su ognuna chiamati Agenti
- * Network based --> controllano punti strategici di una rete tramite dei
sensori (HW) sparsi per la rete e analizzati da un
server chiamato Management System

E in due ulteriori sottotipi:

- * Misuse Based --> come per gli antivirus con le Signature ma non per
il malware ma per "cose strane" nel sistema basandosi
su eventi già noti e studiati --> NON vanno bene per
gli Zero Day
- * Anomaly Based --> si basano sul sapere il comportamento "normale" di
un sistema e controllano per deviazioni dalla norma

In ogni caso l'IDS AVVISA SOLO senza prendere decisioni sul da farsi

Gli Host Anomaly Based usano i file di LOG per analizzare il funzionamento
normale del sistema (training). Al termine della profilazione il sistema sa

moderare l'uso della macchina

I Network Anomaly Based controllano e si addestrano sull'uso normale della rete

Gli Anomaly Based generano molti FALSI POSITIVI

Allarmi totali in organizzazioni grosse --> decine di k

Veri Positivi --> meno dell'1%

Tempo di lavorazione per allarme --> fino a 40 minuti

SIEM

Tutti questi sistemi generano moli titaniche di dati, impossibili da gestire per un umano, ma non per una macchina --> le SIEM Security Information and Event Manager

Analizzano i dati e estraggono gli allarmi più probabili attraverso complicati sistemi di elaborazione di d'allarme, LOG, errori, ...

Sono i cuori dei Cybersecurity Security Operation Centers CSOC --> centri di calcolo giganteschi che lavorano 24/7 per analizzare il traffico

| -> ogni ISP ne ha uno

| -> tentano di prevenire attacchi

| -> analizzano le vulnerabilità

Scartano i falsi positivi

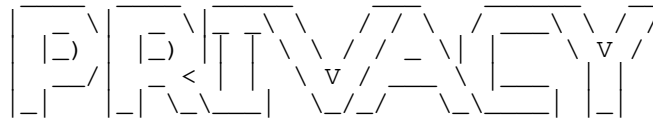
Inoltrano i positivi agli analisti (junior e senior)

* se i junior non sanno analizzarlo vanno ai senior

* i senior confermano o meno l'allarme e nel caso danno l'ordine ai CSIRT (Computer Security Incident response Team, prima CERT) di agire

* gli CSIRT si mettono al lavoro

Un esempio di SOC è WAZUH



COSA E' LA PRIVACY

- * Il diritto a essere lasciati in pace
- * Il diritto alla armonia personale
- * Il diritto ad una vita privata
- * Il diritto al controllo delle informazioni personali
- * Il diritto alla segretezza
- * Il diritto alla anonimità

Un DIRITTO a (vari)

Il concetto nasce nel 1890 da due avvocati (Warren & Brandeis) che stavano difendendo un attore che voleva difendersi da un giornale che ha pubblicato una sua foto paparazzata.

I due avvocati scrissero un libretto di 70 pagine sul concetto della necessità di difesa dalle nuove tecnologie (le macchine fotografiche ai tempi) nella vita privata o come dissero loro: THE RIGHT TO BE LEFT ALONE

Si è evoluto nel corso della storia. Nel 2004 Rodotà (primo Garante della Privacy italiano) che disse che la privacy non è solo il diritto ad essere lasciato in pace ma anche il diritto alla dignità e alla libertà delle persone. Disse inoltre che la privacy è analoga alla libertà di parola e associazione garantita dalla Costituzione in quanto attraverso una violazione di essa una persona può essere discriminata.

Bisogna difendere il "corpo digitale" come quello fisico

PRIVACY != CONFIDENTIALITY

La privacy diventa per Westin diventa un diritto sociale ed è necessario che venga tutelata perché una riduzione può limitare o ledere le libertà personali. Ad esempio tramite ricatto diretto ("se non fai X allora ...") o indiretto (pressione della opinione pubblica su un giurato).

La privacy sis contra con alcune cose come:

- * la prevenzione e punizione del crimine
- * il combattimento al terrorismo
- * il combattere la disinformazione

La privacy è minacciata anche dal valore intrinseco delle informazioni per il marketing e per la pubblicità.

Nel 2026 si stima un valore TOTALE del mercato delle informazioni (Business Intelligence Market) di 55 miliardi di dollari.

Quest'anno si stimano 6 trilardi di dollari di guadagni per l'E-Commerce.

DIGITAL PRIVACY

Il diritto di chiunque di determinare come e quanto le loro informazioni possono essere comunicate a terzi

-Alan Westin

4 tipi:

- * PII Personal Identifiable Information (nome, cognome, codice fiscale, numero di telefono, ...)
- * Non-PII tendenze e inclinazioni di una persona (libri che vengono letti, musica ascoltata, opinioni politiche, ...)
- * Location Data i luoghi fisici
- * Device/Network Data informazioni sulla tecnologia usata (IP, MAC, ..)

PROFILING INFORMATION

File caricati e scaricati, queries di ricerca, comunicazioni, media consumati, siti visitati, interazioni social, commercio elettronico, ...

↑
|

sono già conosciute dai provider dei servizi usati (Google, Vodafone, Spotify, Facebook, Amazon, ...)

PRIVACY THREAT

Una qualunque circostanza che minaccia il controllo di un individuo sulle informazioni che lo riguardano.

Possono essere subliminali (cercare su Google qualcosa mostra già un nostro interesse nella materia cercata) o reali (attacchi di data-breach, furto di identità, ...).

Qui si collega la Sicurezza Informatica e la Privacy. Difendere il sistema per difendere l'utente.

ATTACCHI ALLA PRIVACY

* Furto di Identità---,
 * Profilazione -----| legati a
 * Data Breach <-----'

IDENTITY THEFT

Rubare l'identità a qualcuno e spacciarsi per lui con altri.

Spesso i dati si ottengono attraverso:

- + Phishing
- + Social Engineering
- + Dumpster Diving
- + DATA BREACHES
- + Malware
- + Skimming

I danni più comuni sono:

- + Perdite monetarie
- + Conseguenze legali
- + Danni d'immagine
- + Danni psicologici

PROFILING

Mira a creare un profilo psicologico, abitudinale e comportamentale di qualcuno con lo scopo di influenzare la sua esperienza di shopping o le sue opinioni.

Spesso non bisogna nemmeno "stancarsi" per ottenere i dati, basta offrire un buon servizio e chiedere in cambio informazioni anziché soldi ("tanto è gratis")

DATA BREACH

Attacco ad infrastrutture o contenitori di informazioni sensibili per vendita o sfruttamento diretto.

Possono essere svolti in svariati modi con anche le modalità nella sezione di sicurezza offensiva precedente.

Il Data Breach più famoso fu fatto da Edward Snowden.

Altri importanti furono Cambridge Analitica e Ashley Madison

SOCIETÀ DELLA SORVEGLIANZA

Tutte le attività sono loggate e monitorate tramite il nostro uso quotidiano e quando non collaboriamo attivamente ci pensa il resto della infrastruttura di difesa (CCTV,....)

Combinando tutte le nostre informazioni si può ricreare una copia 1:1 di noi stessi.

Si aggiungeranno in futuro problemi ulteriori sulla parte biometrica

THE AGE OF SURVEILLANCE CAPITALISM

COME SI TENTA DI PROTEGGERE LA PRIVACY

Lato LEGALE:

La legge è ovviamente arrivata dopo i primi fattacci.
L'approccio legale è tipicamente europeo mentre quello statunitense è più "flessibile" e si basa sulla autoregolazione da parte delle aziende, ora stanno pian piano muovendo nella direzione europea, ma per ora solo a livello statale. In EU si usano LEGGI COMPRENSIVE come il GDPR

Lato UTENTE:

Evitare la diffusione di informazioni tramite alcuni step:

- * Privacy Enhancing Technology - ad esempio PGP
- * Self Regulation - evitare di raccontare i fatti propri
- * Privacy Education - spiegare come non raccontare i fatti propri

Lato PROGRAMMATTORE:

- * Anonymity - renderebbe superfluo il resto ma non è sicuro
- * Pseudo-Anonymity - si è noti solo in alcuni contesti
- * Unobservability - non si può essere visti mentre si usa
- * Unlinkability - non si può collegare l'uso all'utente

GDPR

General Data Protection Regulation --> è una direttiva, DEVE essere seguita.
Prevede pene pari al 4% del fatturato annuo o 20 milioni di euro (il più grande)
Ispirato dalle leggi sulla privacy italiane (presenti dal '94)
Può anche generare casi di indagini penali

Detta anche determinati diritti:

- * il diritto all'accesso
- * il diritto all'oblio
- * il diritto allo spostamento dei dati
- * il diritto ad essere informati all'aggiunta dei dati
- * il diritto a far correggere le informazioni
- * il diritto a limitare la processazione dei dati
- * il diritto alla obiezione
- * il diritto alla notifica

Detta cosa è informazione personale:

- * nome
- * indirizzo
- * telefono
- * mail
- * COOKIE
- * IP

Le organizzazioni sono OBBLIGATE ad informare precisamente le informazioni che vogliono raccogliere e come le vorranno usare
C'è dibattito sulla significatività del CONSENSO

NON VIETA la raccolta, ma è regolamentata, deve essere annunciata all'utente e deve essere adeguatamente protetta

In caso di Data Breach l'organizzazione è obbligata ad avvisare il Garante Della Privacy entro 24 ore e da lì parte una indagine per verificare l'adeguata protezione dei dati

DEPERSONALIZZAZIONE

Raccoglimento di informazioni NON riconducibili ad una persona

PRIVACY ENHANCING TECHNOLOGY

Nome collettivo per software e tecnologie che aiutano ad aumentare la privacy dell'utente. Ad esempio:

- * Tor (The Onion Router)
- * TEE (Trusted Execution Environment) - fa in modo che il codice eseguito ed i dati acceduti siano in un ambiente sicuro e protetto in confidenzialità e integrità --> Netflix su telefono ad esempio
- * End2End --> WhatsApp
- * Proxy - "passaggio" attraverso il quale la connessione transita che nasconde la connessione originale

A ----> Proxy ----> B
??

Terzo vede Proxy verso B, non A verso B
Può essere lento

Se uno lo "rompe" può vedere A -> B

TOR

Creato dal DARPA e dalla US Naval Intelligence

E' un "proxy multiplo" --> sistema la problematica del "singolo nodo" del PROXY normale

- > sono svariati nodi tra me e la fonte
- > le comunicazioni tra utente e nodo, nodo e nodo e nodo server sono criptate --> A CIPOLLA

|
v

su tre livelli --> 3 inizializzazioni con scambio chiavi tra me e nodo

: -) [[[P]]] -> A [[P]] -> B [P] -> C P -> Server
 DA C A SERVER SE SI USA HTTP E' IN CHIARO
 TOR BROWSER FORZA HTTPS PER OVVIARE

P : -) [[[P]]] <- A [[P]] <- B [P] <- C P <- Server P

Se il sito è .onion i nodi di ritorno NON sono A,B e C ma J, H e G

- > gli URL sono alfanumerici e non sono facilmente memorizzabili da un umano --> spesso raccolti in elenchi tipo quelli del telefono e potrebbero non funzionare più
- > i nodi sono sia server sia altri utenti
- > i nodi possono essere non VICINI
- > i nodi rallentano la connessione
- > è possibile fare attacchi di correlazione al 1° e 3° nodo