

## WiNpcap - CMEDDler

Maurizio De Pascale - 056/101555  
Salvatore P'Amico - 056/001186

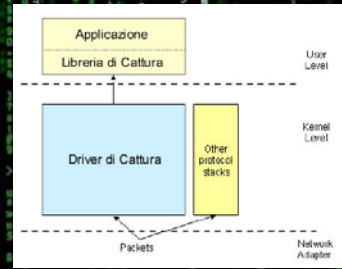
## Road Map

- Cattura di Pacchetti
  - Introduzione
  - Architettura
- CMEDDler

## Cattura dei Pacchetti

- Cos'è
- Come funziona
  - Hardware
  - Software
- Utilizzi
  - Analisi e monitoraggio della rete
  - ...

## Architettura di cattura

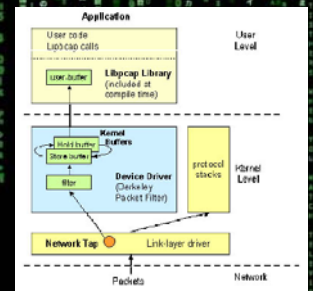


## Architettura di cattura

- Unix - Berkeley Packet Filter
- Windows - WinPcap

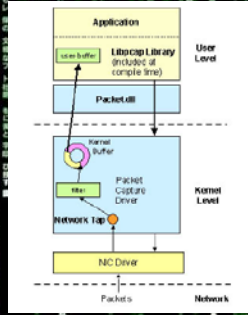
## Unix

- Unix - Berkeley Packet Filter
  - Pseudo Device
  - LibPcap



## Windows

- WinPCap
- Driver di cattura
- Packet.dll
- WinPCap.dll



## Processo di filtraggio

- cose e
- Perché
- come
- Hardware
- Software

## Filtraggio

- Hardware
- Modalità adattativa (Promiscua - Broadcast - Multicast...)
- Software
- Macchina virtuale
- BPF Code
- BPF Assembler

## BPF Pseudo machine

- Perché
- Sicurezza e Adattabilità
- Implementazione
- Accumulatore A
- Registro indice X
- Memoria Temporanea M[]
- Pacchetto P[]

## Packet.dll

- Inizializzazione
- Lettura dei Pacchetti
- Scrittura dei Pacchetti
- Statistiche ed informazioni
- Rilascio delle risorse

## Packet.dll - Inizializzazione

- PacketGetAdapterNames
- PacketOpenAdapter
- PacketAllocatePacket
- PacketInitPacket
- PacketSetWFILTER
- PacketSetUFF
- PacketSetBPF

## PacketDill - Lettura

- PacketReceiverPacket
- PacketSetReadTimeout
- PacketSetMode

## PacketDill - Scrittura

- PacketSendPacket
- PacketSendWrites

## PacketDill - Statistiche ed info

- PacketStats
- PacketEnctype

## PacketDill - Rilascio delle risorse

- PacketCloseAdapter
- PacketReepacket

## CMEDDler

- Csniffer - Cinjector
- BPFAssembler
- NetworkSmartPointer

## Csniffer - Cinjector

- Perché
  - Possibili collocazioni nell'architettura
  - semplicità VS potenza

## Vantaggi

- Programmazione a oggetti
- Nomenclatura Windows
- Gestione automatica delle risorse
- Gestione automatica dell'I/O
- Uniformità nella gestione degli errori
- Nessun overhead

## CMEddler - Esempio

```
Spicche.GetAdaptersList(List);  
Spicche.Open(List.nameOfAdapter[index]);  
Spicche.SetDriverBuffer(2000000);  
Spicche.SetHardwareFilter(NDIS_PACKET_TYPE_PROMISCUOUS);  
Spicche.InitBuffer(1000000);  
  
while(!kbhit())  
  
    Spicche.ReceivePacket(pHeader, (PBYTE&)Frame);  
    PrintPacket(pHeader,Frame);
```

## CMEddler - Metodi comuni

### Enumerazione Apertura e Chiusura

```
VOID GetAdaptersList(ADAPTERS_LIST&  
    adapterList);  
BOOL32 Open(PCHAR8 adapterName);  
VOID Close();
```

## CMEddler - Metodi sniffer

### Inizializzazione risorse

```
BOOL32 InitBuffer(UINT32 bufferSize);  
BOOL32 SetDriverBuffer(INT32 bufferSize);  
BOOL32 SetHardwareFilter(ULONG filter);  
BOOL32 SetBpfFilter(BPF_PROGRAM& filter);
```

## CMEddler - Metodi sniffer (2)

### Statistiche ed Informazioni

```
BOOL32 SetMode(INT32 mode);  
BOOL32 GetStats(ADAPTER_STATS& stats);  
BOOL32 GetNetType(NET_TYPE& netType);
```

## CMEddler - Metodi sniffer (3)

### Letture

```
BOOL32 SetTimeout(INT32 timeout);  
BOOL32 SetMinToCopy(INT32 bytes);  
BOOL32 ReceivePacket(BPF_HEADER_PTR&  
    pHeader, PBYTE& pData);
```

### Rilascio Risorse

```
BOOL32 Reset();  
VOID FreeBuffer();
```

## C M eDdlEr - M etoD; Injector

### Scrittura

```
BOOL32 SendPacket(PBYTE pFrame, UINT32  
size);  
BOOL32 SetNumberOfWrites(INUINT32 writes);
```

## B P F A sseMBlEr

- **P**roblematiche con i Filtri
  - Semplicità vs Potenza
- **C**os'è
  - Generazione macro
  - Generazione codice
- **C**ome si usa

## B P F A sseMBlEr - E sempIo

```
! Dh [12] //Carica il tipo ethernet  
jeq #0x800,0,5 //se non e' IP rifiuta il pacchetto  
! Dh [20] //carica il fragment offset & flags  
jset #0x1fff,3,0 //salta se fragment offset non e' 0  
! D B [23] //carica il Protocollo dall'header IP  
jeq #0x6,0,1 //salta se non e' tcp  
ret #-1 //accetta tutto  
ret #0 //rifiuta il pacchetto
```

## B P F A sseMBlEr - E sempIo macro

```
BPF_STMT(BPF_LD+BPF_H+BPF_ABS,12),  
BPF_JUMP(BPF_JMP+BPF_JEQ+BPF_K,0x800,0,5),  
BPF_STMT(BPF_LD+BPF_H+BPF_ABS,20),  
BPF_JUMP(BPF_JMP+BPF_JSET+BPF_K,0x1fff,3,0),  
BPF_STMT(BPF_LD+BPF_B+BPF_ABS,23),  
BPF_JUMP(BPF_JMP+BPF_JEQ+BPF_K,0x6,0,1),  
BPF_STMT(BPF_RET+BPF_K,-1),  
BPF_STMT(BPF_RET+BPF_K,0),
```

## B P F A sseMBlEr u tilizzo

```
C:\>bpfasm -ifilter_IP.txt -cfilter_ip.bpf
```

```
C:\Sniffer Spione;  
BPF_PROGRAM Filter;  
  
if (Filter.Load("filter_IP.bpf"))  
{  
printf("Filter loaded...\n");  
if (! Spione.SetBpfFilter(Filter))  
printf("Invalid filter...\n");  
}  
else printf("Unable to load filter...\n");
```

## N etwork smArT P ointer

- **P**roblematiche
  - conoscenza approfondita dei Protocolli
  - Byte ordering
  - conversioni di formata

## NSP Esempio

```
BYTE buffer;  
EthernetFramePtr Frame;  
IP4DatagramPtr Datagram;  
Frame=buffer;  
  
If (Frame.Type()==0x0800)  
{  
    Datagram=Frame.Data();  
}
```

## NSP Implementazione

```
NAKED VOID CEthernetFrame::Type(UINT16 type)  
{  
    __asm  
    {  
        mov ax,[esp+4] //mette in AX type  
        rol ax,8 //lo inverte  
        mov ecx,[ecx] //carica pData in ECX  
        mov [ecx+ETH_TYPE_OFFSET],ax  
        ret 4 //ritorna e pulisce lo stack  
    }  
}
```

## IP 4 Datagram

```
UINT8 Version();  
UINT8 HeaderLength();  
UINT8 TypeOfService();  
UINT16 TotalLength();  
UINT16 Identification();  
UINT8 Flags();  
UINT16 FragmentOffset();  
UINT8 TimeToLive();  
UINT8 Protocol();  
UINT32 SourceAddress();  
UINT32 DestinationAddress();  
UINT16 Checksum();  
PBYTE Options();  
PBYTE Data();  
  
VOID Version(UINT8 version);  
VOID HeaderLength(UINT8 length);  
VOID TypeOfService(UINT8 service);  
VOID TotalLength(UINT16 length);  
VOID Identification(UINT16 id);  
VOID Flags(UINT8 flags);  
VOID FragmentOffset(UINT16 offset);  
VOID TimeToLive(UINT8 time);  
VOID Protocol(UINT8 protocol);  
VOID SourceAddress(UINT32 address);  
VOID DestinationAddress(UINT32 address);  
VOID ComputeChecksum();
```