Sicurezza delle reti[1]

Mattia Monga

Dip. di Informatica
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

a.a. 2012/13

Lezione XI: Risposta

Risposta di un NIDS

- La tipica risposta di un NIDS al verificarsi di un evento che verifica una firma è la generazione di un allarme
- La forma piú standard di allarme è la scrittura in un corrispondente file di log

Risposta di un NIDS

```
[1:1122:2 ] WEBMISC /etc/passwd [Classification: Attempted
Information Leak ] [Priority:2 ] 09/1610:04:15.826116
192.168.1.1:3143 >192.168.1.2:80 TCP TTL:128 TOS:0x0
ID:12832 IpLen:20 DgmLen:149 DF ***AP***Seq:0xDEFF5454
Ack:0x1A51AF74 Win:0x4470
```

Esistono molte varianti implementate dai diversi NIDS, tra cui salvataggio in formato tcpdump, scrittura su database (es. MySQL), visualizzazione a video ecc.

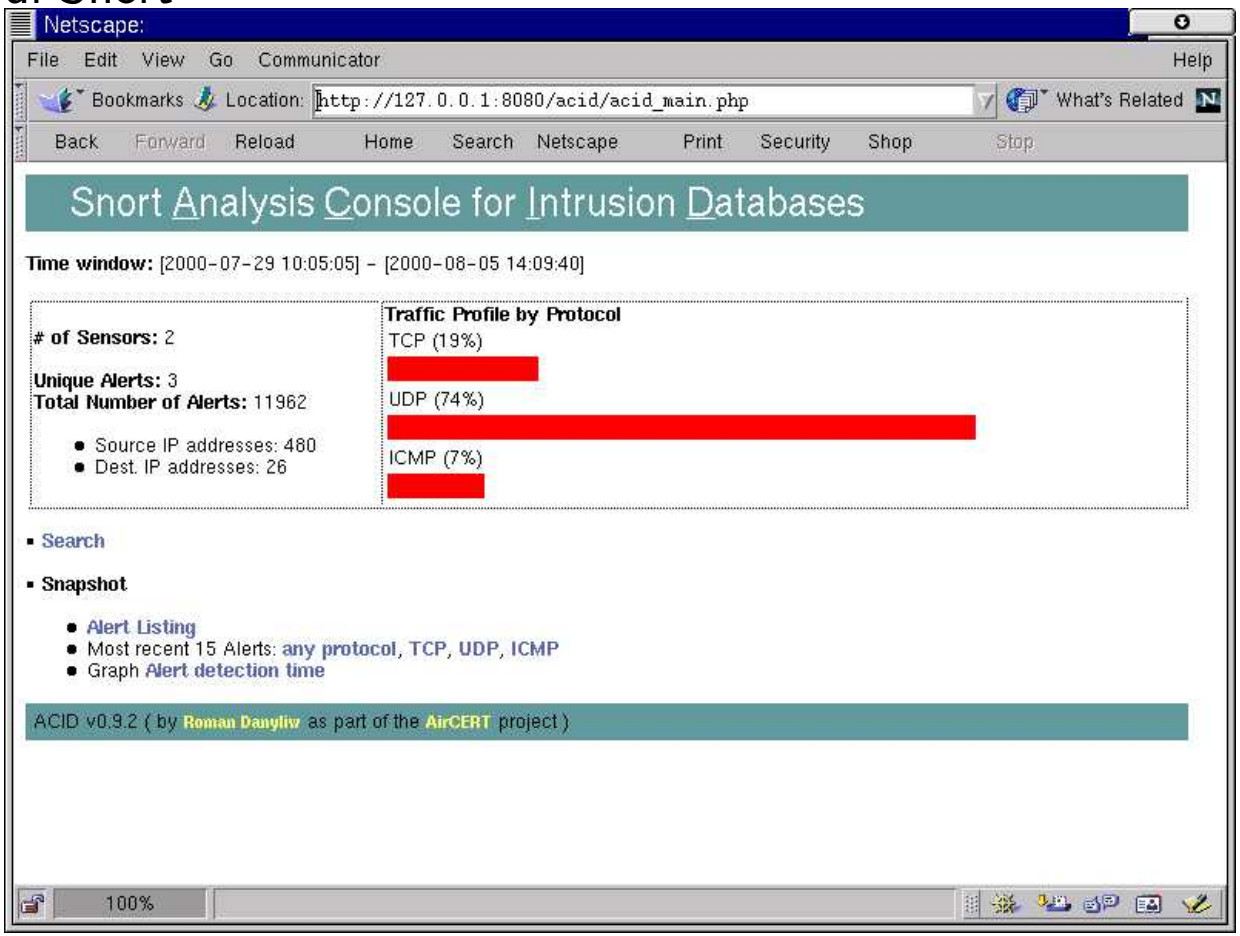
Analisi

La mole di dati è imponente.

- Esistono molti strumenti, sia open-source che integrati nei prodotti commerciali, di analisi dei log prodotti da un NIDS.
- Tipicamente vengono mostrati grafici, statistiche ecc. Sono utili per le analisi *post-mortem* e per il tuning dei sistemi, ma inefficaci per un'azione di contenimento real-time
- L'invio di email a un amministratore è un'altra modalità di risposta diffusa (e onerosa).

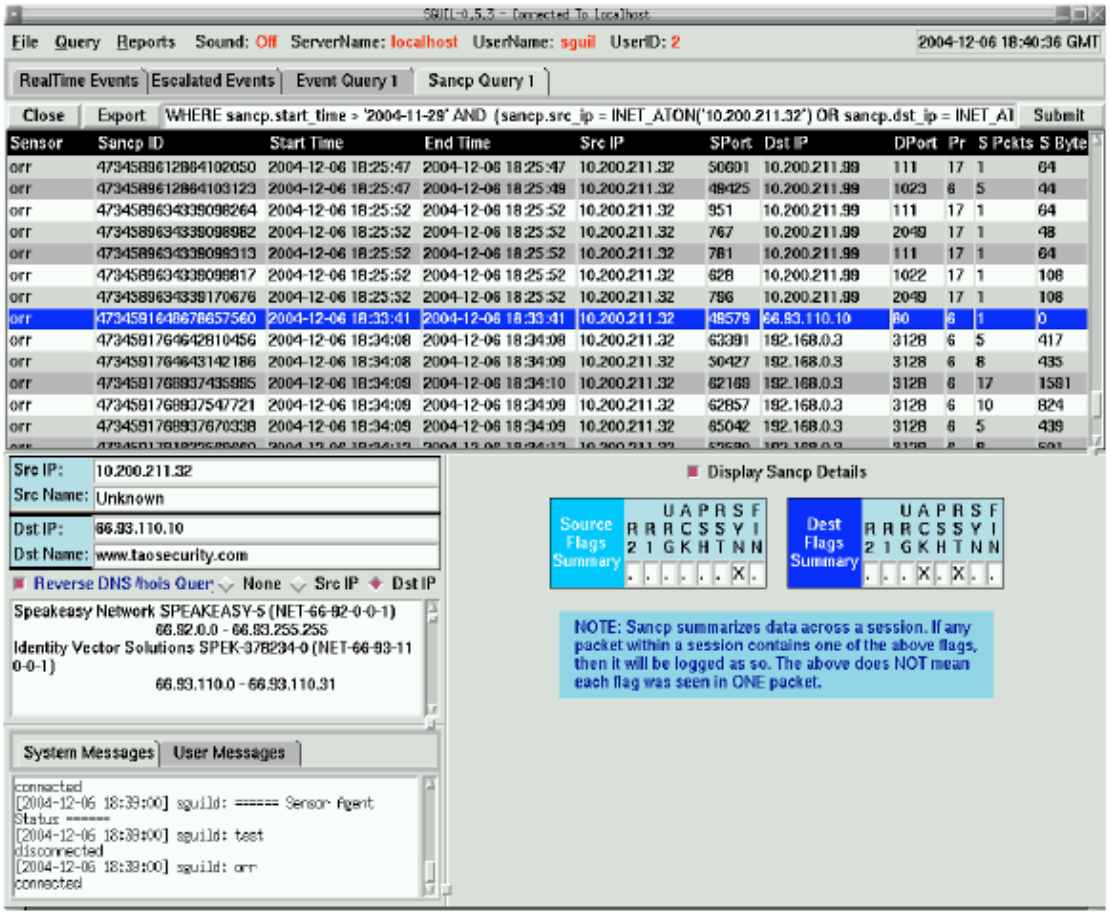
Tool di analisi

ACID (Analysis Console for Intrusion Databases)
http://acidlab.sourceforge.net/ Interfaccia in PHP di analisi dei log di Snort



Tool di analisi

SGUIL (The Analyst Console for Network Security Monitoring) http://sguil.sourceforge.net/index.php Interfaccia per la visualizzazione real-time di alarm generati da Snort

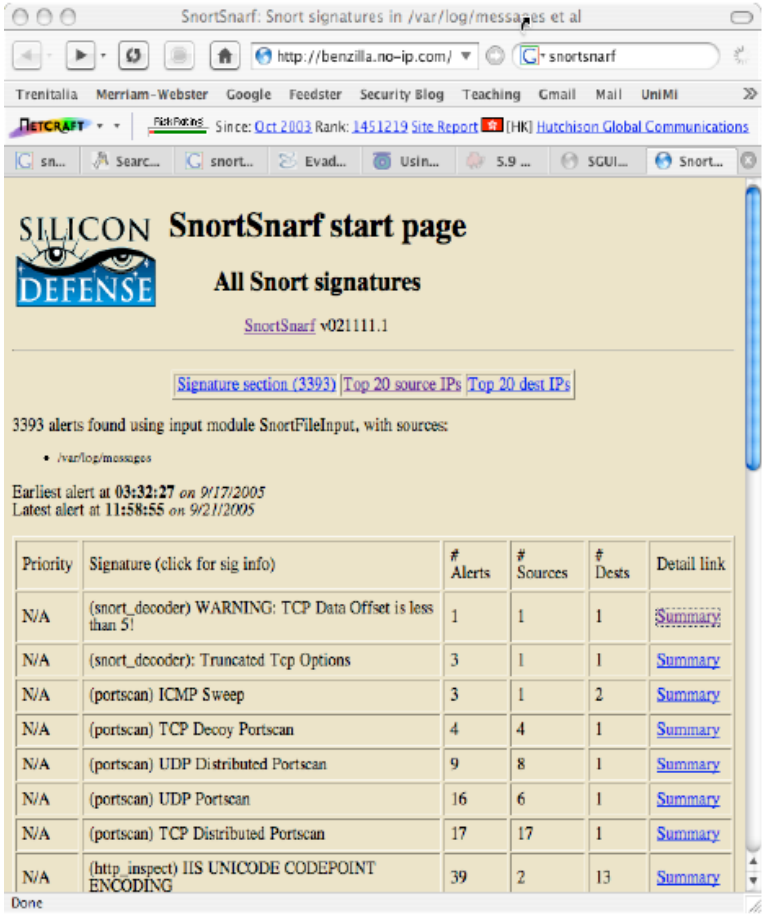


Tool di analisi

SNORTSNARF

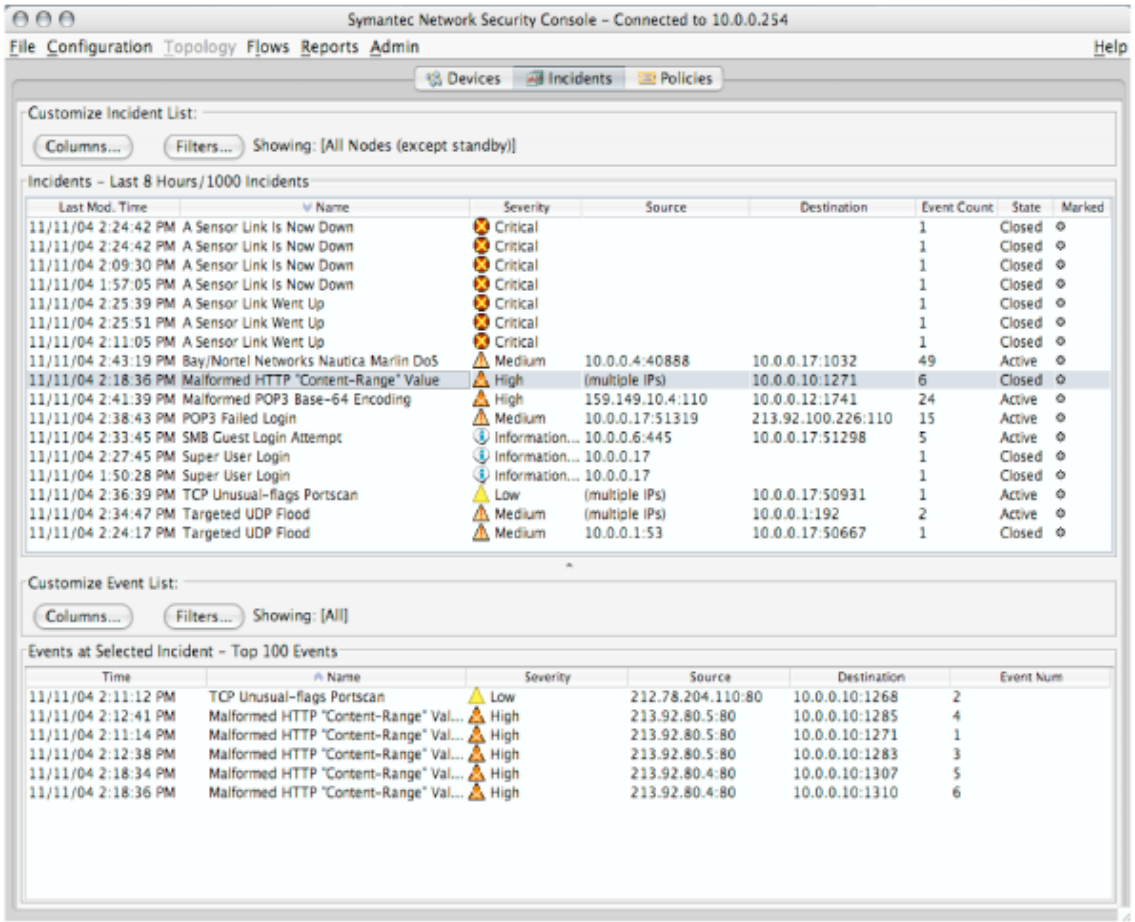
http://www.snort.org/dl/contrib/data_analysis/snortsnarf/

Interfaccia WEB per l'analisi dei log generati da Snort



Tool di analisi

Symantec Network Security 7120 Interfaccia per l'analisi dei log generati dall'appliance



Risposte automatiche

Risposta automatica

Una modalità di allarme che implica la generazione automatica di azioni allo scopo di rispondere attivamente ad una presunta intrusione senza richiedere l'intervento diretto di un operatore.

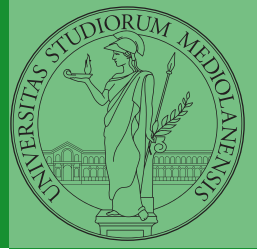
Risposte automatiche

Esempio Snort:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-IIS cmd.exe access"; content:"cmd.exe";
react: block; ...)
```

L'opzione `react: block` fa sí che la connessione TCP nella quale si è verificato il tentativo di accesso a `cmd.exe` venga automaticamente terminata

Tipologia di risposte automatiche



Le tecniche piú diffuse sono:

- Reset di sessioni (Session Sniping)
  - L'esempio precedente con Snort è di questo tipo
- Aggiornamento del firewall

Reset di sessioni

Per lo sniping, il NIDS deve essere in grado di forzare la terminazione della connessione

- inviando un pacchetto contenente un RST a entrambi
- devono apparire ai riceventi come inviati dalle controparti

Aggiornamento dei firewall

La rilevazione di un allarme può essere sfruttata per riconfigurare automaticamente le regole di un firewall

- Esempio: la rilevazione di attività di scan viene utilizzata per impedire automaticamente ogni connessione da parte degli indirizzi IP sorgenti coinvolti.

Aggiornamento dei firewall

Meno efficace di quel che potrebbe sembrare:

- Un intrusore può provocare riconfigurazioni che risultano dannose, ad esempio inviando pacchetti con IP spoofed
- Gli effetti possono essere di bloccare le connessioni provenienti da sorgenti legittime (denial-of-service)

Riassumendo

Cosa fare delle segnalazioni dell'IDS

- usare tool di analisi
- interrompere connessioni
- riconfigurare, piú o meno automaticamente, le regole dei firewall

Evasione di IDS

Spesso l'elusione del rilevamento è possibile sfruttando l'uso di alias o altri trucchi che aggirano l'identificazione di una risorsa o di un attacco

Evasione di IDS

Esempio: Una regola che cerchi di verificare la condizione `content:/etc/passwd;` potrebbe essere bypassata da formati equivalenti quali `/etc//\//passwd` oppure `/etc/rc.d/.././\passwd`.
Occorre cercare di riportare la regola all'esame di nomi canonici.

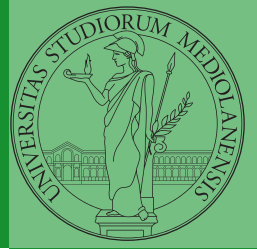
Evasione di IDS

Tecniche di evasione piú sofisticate utilizzano pacchetti frammentati per la loro difficoltà di gestione.
Per esempio, si supponga che il NIDS abbia una finestra per riassemblare i pacchetti frammentati inferiore rispetto al sistema vittima. Il NIDS considererebbe due frammenti come pacchetti indipendenti, il sistema destinatario come pacchetto unico.

Pericoli delle risposte automatiche

Qualunque meccanismo di risposta automatica ha il potenziale difetto di poter essere bypassato e/o sfruttato contro il sistema stesso che viene protetto

Costi delle risposte automatiche



- Le risposte automatiche non sostituiscono l'intervento e l'analisi dell'operatore umano: un apparente risparmio di risorse può risultare in un aggravio di costi
- L'intrusion detection è per sua natura un'attività che necessariamente richiede una forte componente di analisi e di gestione manuale da parte di operatori specializzati (per questo è spesso esternalizzato).

Commenti generali sugli IDS

Un famoso (e controverso) rapporto di Gartner Group del 2003 afferma che gli IDS non valgono gli investimenti richiesti, perché:

- Troppi falsi positivi e negativi
- Richiedono staff dedicato al monitoraggio che dev'essere compiuto 24×7

Commenti generali sugli IDS

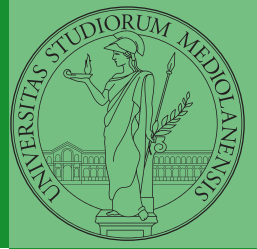
- Il processo di risposta agli incidenti è molto oneroso
- Non si riescono a monitorare reti con traffico superiore ai 600MB/s senza inaccettabili decadimenti prestazionali

Commercialmente si è passati al termine IPS (intrusion protection s.), suggerendo cosí di avere a che fare con strumenti piú sofisticati...

Riassumendo

- Le risposte automatiche hanno costi organizzativi e possono risultare strumenti di evasione o attacco
- Il processo di risposta agli incidenti è molto oneroso e richiede staff esperto

Attacchi imprevisti



I NIDS signature-based si basano sull'assunzione di saper caratterizzare un attacco.

1. Identificare una vulnerabilità: la firma cercherà di rappresentare tutti gli attacchi capaci di sollecitarla;
2. Riconoscere un exploit: la firma cercherà di rappresentare tutte le varianti.

Attacchi imprevisti

Zero day

Un attacco può essere del tutto inatteso: in questo caso si parla di zero-day, ossia il giorno *prima* di quando i NIDS sono in grado di riconoscerlo.

Finestra di vulnerabilità

Il tempo che intercorre fra il momento in cui un attaccante si rende conto di una vulnerabilità e capisce come sfruttarla e il momento in cui l'attacco è identificato dal difensore può essere molto lungo (*vulnerability window*).

Nel 2008 Microsoft ha reso nota una vulnerabilità di IE presente dal 2001, quindi con una finestra potenzialmente di 7 anni!

Ricerca delle vulnerabilità

La ricerca delle vulnerabilità non note è una delle attività dei "laboratori di sicurezza".

- Si cercano vulnerabilità generiche (non di una rete specifica): si analizzano applicazioni e protocolli
- Gli *zero-day* hanno un mercato (non solo underground!)

Tecniche per la ricerca

studio analitico si studiano le specifiche piú o meno formali di applicazioni e protocolli

fuzzing si provano le applicazioni (o i protocolli) con input "strani" casuali

honeypot un sistema che viene realizzato e messo in opera solo come bersaglio

Polimorfismo degli attacchi

La medesima vulnerabilità può essere sfruttata da exploit con forme diverse: gli attacchi hanno quindi natura polimorfica.
In generale è impossibile prevedere tutte le possibili varianti e costruire le firme che permettano di rilevarli.
Una forma completamente nuova è analoga a uno zero-day, anche se la vulnerabilità è già nota.

Cifratura

Una delle tecniche piú diffuse è la cifratura.

- Viene generata per ogni attacco una chiave casuale
- il *payload* dell'attacco viene cifrato, apparendo cosí sempre diverso
- l'unica parte di codice costante è una piccola routine di decifratura (possono bastare 3-4 istruzioni: p.es. cifratura XOR)
- anche la routine di decifratura può essere variata con ulteriori tecniche di polimorfismo

Dead-code insertion

dead-code insertion o trash insertion: aggiungere codice senza modificare il comportamento.

- La tecnica piú semplice è inserire `nop`
- Metodi piú sofisticati fanno uso di sequenze di codice che si annullano vicendevolmente

La ricerca di stringhe costanti fallisce.

```
call 0h
pop ebx
lea ecx, [ebx + 45h]
nop
nop
push ecx
push eax
inc eax
push eax
dec [esp - 0h]
dec eax
sidt [esp - 02h]
pop ebx
add ebx, 1Ch
cli
mov ebp, [ebx]
```

Code transposition

Sposta le istruzioni in modo che l'ordine del codice binario sia differente dall'ordine di esecuzione

- riordinando casualmente blocchi di istruzioni e inserendo salti incondizionati (facile da fare automaticamente)
- mischiando istruzioni indipendenti (richiede analisi sofisticate del codice)

```
call 0h
pop ebx
jmp Step2
Step3: push eax
push eax
sidt [esp - 02h]
jmp Step4
add ebx, 1Ch
jmp Step6
Step2: lea ecx, [ebx + 45h]
push ecx
jmp Step3
Step4: pop ebx
cli
jmp Step5
Step5: mov ebp, [ebx]
```

Instruction substitution e register reassignment

- instruction substitution: dizionari di sequenze di istruzioni equivalenti, che possono essere sostituite tra loro.
- register reassignment sostituisce l'uso di un registro con un altro equivalente.

```
call 0h
pop ebx
lea ecx, [ebx + 42h]
sub esp, 03h
sidt [esp - 02h]
add [esp], 1Ch
mov ebx, [esp]
inc esp
cli
mov ebp, [ebx]
```

Riassumendo

Gli IDS misuse-based necessitano di *firme* degli attacchi:

- A volte non sono ancora note
- È difficile prevedere le varianti introdotte con tecniche di polimorfismo

Generatori di signature

L'idea di base è che grazie alla conoscenza di vulnerabilità e di un certo numero di exploit, si vogliono generare automaticamente signature utili a bloccare exploit non ancora rilevati "in the wild".

Generatori di signature

semantic-based modellano il comportamento di un attacco: se la rilevazione richiede l'interpretazione del modello, può essere molto dispendiosa.

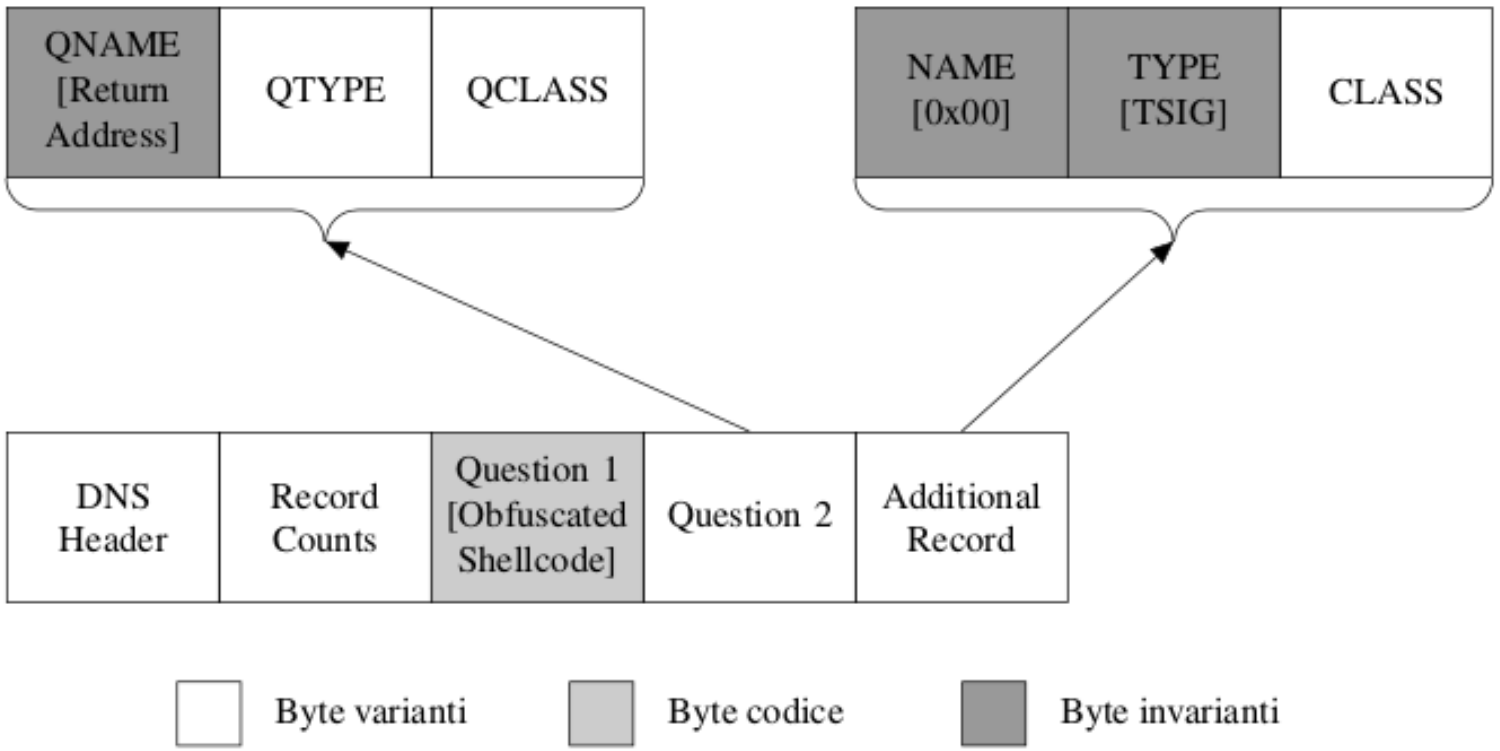
content-based si basano sulla ricerca di invarianti: in realtà è piuttosto raro che la parte invariante di un exploit sia sufficientemente ricca per limitare i falsi positivi.

Hamsa

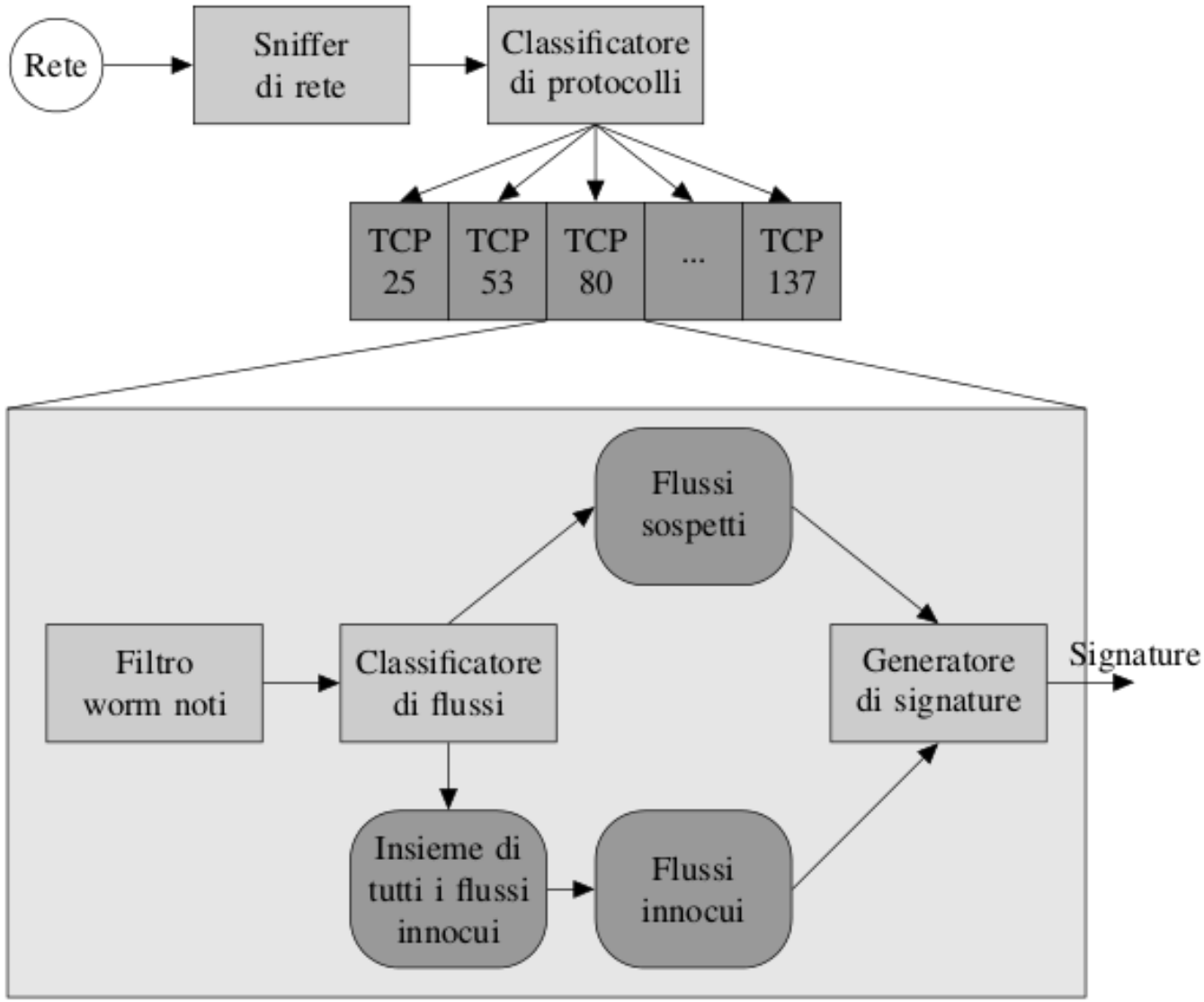
- Zhichun Li, Manan Sanghi, Yan Chen, Ming-Yang Kao and Brian Chavez. Hamsa: Fast Signature Generation for Zero-day Polymorphic Worms with Provable Attack Resilience. IEEE Symposium on Security and Privacy, Oakland, CA, maggio 2006.
- Utilizzabile a livello di rete (gateway e router)
- *content-based*:

invarianti	byte il cui valore e fissato a priori e la cui variazione implica il fallimento dell'attacco
code byte	parte potenzialmente polimorfica, ma con una semantica fissa
wildcard byte	possono assumere qualsiasi valore

Esempio: Lion worm



Architettura di Hamsa



Architettura di Hamsa

classificatore di protocolli considera i flusso TCP (o i pacchetti UDP) e li classifica secondo la porta destinazione

politica di selezione indica quali flussi prelevare e inviare al generatore di signature

generatore di signature genera le signature, considerando i flussi innocui e sospetti

Firme Hamsa

- Sono dette conjuction signature: consiste in un insieme di stringhe e un flusso viene considerato malevolo se contiene tutte le stringhe, indipendentemente dall'ordine.
- Si tratta in realtà di *multi-insiemi* di token, cioè insiemi in cui un elemento può apparire piú volte.

Firme Hamsa

Code-Red II	{'.ida?':1, '%u780':1, ' HTTP/1.0\r\n':1,'GET /':1, '%u':2}
ATPhttpd	{'\x9e\xf8':1, ' HTTP/1.1\r\n':1, 'GET /': 1}

I token indicati devono comparire in un unico flusso e con un numero di occorrenze maggiore o uguale a quello indicato.

Algoritmo di generazione delle firme

Input: Insieme degli invarianti I, insieme dei flussi malevoli M e dei flussi innocui N, vettore u dei falsi positivi massimi
Output: Signature S per un worm presente in M

```
S = creaSignatureVuota()
SignatureCandidata = S
VettoreSignature = []
i = 1
while i < k do
    foreach t ∈ I do
        S = S.aggiungi(t)
        FP = calcolaFalsiPositivi(S, N)
        if FP < u[i] then
            TP = calcolaVeriPositivi(S, M)
            if SignatureCandidata.TP < TP then
                SignatureCandidata = S
            end
        end
        S = S.rimuovi(t)
    end
    if SignatureCandidata == creaSignatureVuota() then
        break
    end
    VettoreSignature.appendi(SignatureCandidata)
    S = SignatureCandidata
    SignatureCandidata = creaSignatureVuota()
    i = i + 1
end
foreach S ∈ VettoreSignature do
    calcolaPunteggio(S)
end
return S con punteggio massimo
```

- k è il numero di token in $\mathcal{I}$
- $\text{calcolaPunteggio}(\mathcal{S}) = -log_{10}(\delta + FP_{\mathcal{S}}) + a \cdot TP_{\mathcal{S}} + b \cdot \text{lunghezza}(\mathcal{S})$
- tutti i parametri sono scelti in modo empirico (anche per la classificazione innocuo/sospetto)
- $u(i) = u(1) \cdot {u_r}^{(i-1)}$ con $u(1) = 0{,}15$ e $u_r = 0{,}5$

Attacchi a Hamsa

Target feature manipulation Si cerca di variare le parti considerate invarianti

Innocuous pool poisoning prima di iniziare la diffusione vera e propria e quindi prima di lanciare un attacco verso una nuova macchina, ci si preoccupa di inviare una serie di pacchetti leciti contenenti ciascuno un invariante inserito nelle parti di traffico che possono essere modificate a piacere.

Suspicious pool poisoning l'attaccante incorpora finti invarianti all'interno dei flussi malevoli per portare alla generazione di signature che dipendono da tali finti invarianti al posto o in aggiunta agli invarianti veramente necessari.

Riassumendo

Generare automaticamente le varianti di un attacco:

- È un'operazione con fortissime connotazioni empiriche (in generale è un obiettivo irrealizzabile)
- Come sempre, un meccanismo automatico può essere sfruttato anche dall'attaccante (*poisoning*)