Sicurezza dei sistemi e delle reti[1]

Mattia Monga

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

a.a. 2015/16

[1] 🅭🅯🅭 2011–15 M. Monga. Creative Commons Attribuzione — Condividi allo stesso modo 4.0 Internazionale. http://creativecommons.org/licenses/by-sa/4.0/deed.it. Derivato con permesso da © 2010 M. Cremonini.

Lezione X: 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.

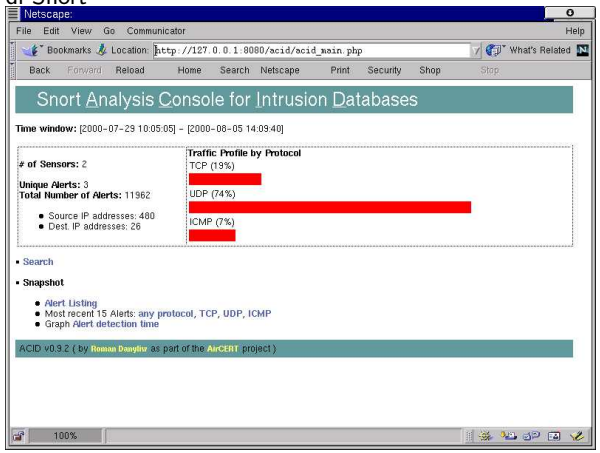
Analisi

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



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

Risposte automatiche

Risposta automatica

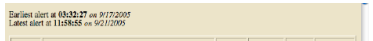
Una modalità di allarme che implica la generazione automatica di azioni allo scopo di rispondere attivamente ad una presunta intrusione senza richiedere l'intervento diretto di un operatore.

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

Buffer overrun/overflow (la tracimazione)

Vulnerabilità classica: *pubblicata* per la prima volta nel 1996 ("Smashing the stack for fun and profit", Aleph1), ma certamente nota ben prima (viene usata nell'Internet worm). Il problema nasce da un accesso ad una zona di memoria non esplicitamente allocata dal programmatore

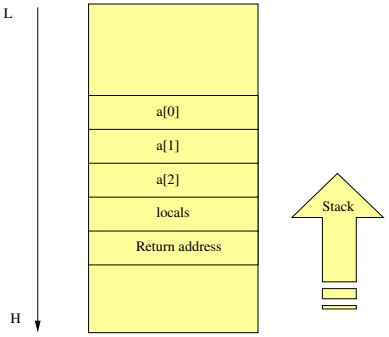
- È di gran lunga il problema piú frequente
- Sorge perché C e C++, per motivi di efficienza, non controllano gli accessi alla memoria

Vulnerabilità antica, eppure gli esperti in grado di seguirne i dettagli sono ancora pochi!!!

`http://10kstudents.eu/`

Il principio dell'attacco

a[i] == &a[0] + **sizeof**(a[0])∗i



Esempio di buffer overrun

```
1  int scanf(const char *format, ...);
2  int printf(const char *format, ...);
3  char *strcpy(char *dest, const char *src);
4
5  void funct(const char stringa[]);
6
7  int main(void) {
8    char tmp[100];
9
10   if (scanf("%s",tmp) > 0) {
11     funct(tmp);
12     return 0;
13   }
14   return -1;
15 }
16
17 void funct(const char stringa[]) {
18   char buffer[3];
19   strcpy(buffer, stringa);
20 }
21
22 void exit(int status);
23
24 void hack(void) {
25   printf("Ho eseguito una funzione senza chiamarla!\n");
26   exit(0);
27 }
```


GDB (1)

```
(gdb) break funct
Breakpoint 1 at 0x80484dd: file bo.c, line 19.
(gdb) run
Starting program: /home/monga/lavoro/Didattica/materiale-sicurezza/bo
12

Breakpoint 1, funct (stringa=0xffffcf3c "12") at bo.c:19
19   strcpy(buffer, stringa);
(gdb) info frame
Stack level 0, frame at 0xffffcf20:
 eip = 0x80484dd in funct (bo.c:19); saved eip = 0x80484c0
 called by frame at 0xffffcfc0
 source language c.
 Arglist at 0xffffcf18, args: stringa=0xffffcf3c "12"
 Locals at 0xffffcf18, Previous frame's sp is 0xffffcf20
 Saved registers:
  ebp at 0xffffcf18, eip at 0xffffcf1c
```


GDB (2)

```
(gdb) print hack
$1 = {void (void)} 0x80484f2 <hack>
(gdb) print (char*)buffer
$2 = 0xffffcf0d ""
(gdb) print 0xffffcf1c - 0xffffcf0d
$3 = 15
(gdb) print/o {0x08,0x04,0x84,0xf2}
$4 = {010, 04, 0204, 0362}
(gdb) set stringa = "123456789012345\362\204\004\010"
(gdb) x/x buffer + 15
0xffffcf1c: 0x080484c0
(gdb) next
20 }
(gdb) x/x buffer + 15
0xffffcf1c: 0x080484f2
(gdb) continue
Continuing.
Ho eseguito una funzione senza chiamarla!
[Inferior 1 (process 7990) exited normally]
```


Ricapitolando: il buffer overrun

- Debolezza: array e aritmetica dei puntatori (C, C++).
- Condizioni di attacco:
  - chiamata a procedura
  - layout della memoria
  - input

Che codice può essere eseguito

Nell'esempio il codice eseguito fa parte del "programma". È però possibile passare nella stringa incriminata

- l'indirizzo di ritorno voluto *nello stack*
- le istruzioni da eseguire (di solito una shell minima: possono bastare 10 caratteri)