Laboratorio di sicurezza informatica

(prima parte)

Gabriele D'Angelo, Ludovico Gardenghi

{gda, garden}@cs.unibo.it

Università di Bologna

Dipartimento di Scienze dell'Informazione

Giugno, 2005

Scaletta della lezione

- Il problema della disclosure
- Internet worm
- Buffer overflow
- Password brute-force attack
- Vulnerabilità del kernel
- Rootkit + Intrusion Detection System
- Port Scanner
- Security Scanner
- Firewall
- Note e bibliografia

Warning!

- Quanto vi presenteremo oggi e nelle prossime lezioni è a puro

  scopo didattico

- Le intrusioni nei sistemi informatici sono punite dalla legge e quindi non è il caso di giocare senza capire bene cosa si sta facendo

- Esistono le macchine e le reti virtuali: usiamole!

- Vi impegnate ad usare quanto appreso a solo scopo difensivo?

Il problema della “disclosure”

- Quando una vulnerabilità viene scoperta e viene scritto un exploit funzionante ci si può comportare in diversi modi.
- Quanto e a chi viene diffusa l'informazione?
  - **full disclosure**: tutti i dettagli sono immediatamente pubblicati
  - **partial disclosure**: i dettagli sono inizialmente comunicati solo agli sviluppatori del software vulnerabile e, dopo qualche tempo, sono resi noti al pubblico
  - **no disclosure**: gli unici informati dell'esistenza e dei dettagli della vulnerabilità sono gli sviluppatori del software
- Un exploit sconosciuto alla comunità o per il quale non esistono ancora contromisure è detto **zero-day exploit**
- C'è un approccio migliore degli altri?

Nota storica: l'Internet Worm

- Il 2 novembre 1988 Internet ha subito la prima diffusione di un worm su larga scala
- Il worm era formato da:
  - Un programma di bootstrap scritto in C (99 linee di codice)
  - Un file oggetto (versioni VAX e SUN-3)
- L'effetto principale del worm era la replicazione e un notevole **aumento del carico**, tale da bloccare le macchine infettate
- La stima dei danni ammonta a 6000 macchine infettate, nessun danno fisico ma un costo stimato del problema estremamente elevato

Nota storica: l'Internet Worm

- Come si propagava il worm:
  - Buffer overflow in fingerd
  - Vulnerabilità in sendmail (attraverso il codice di DEBUG)
  - rsh (+ ricerca di password banali in /etc/passwd)
- Da notare:
  - Il worm **non** è stato scritto con fini maligni
  - Il worm contiene vari errori di programmazione
  - I meccanismi di limitazione delle replicazione non riescono ad impedire infezioni multiple sullo stesso server

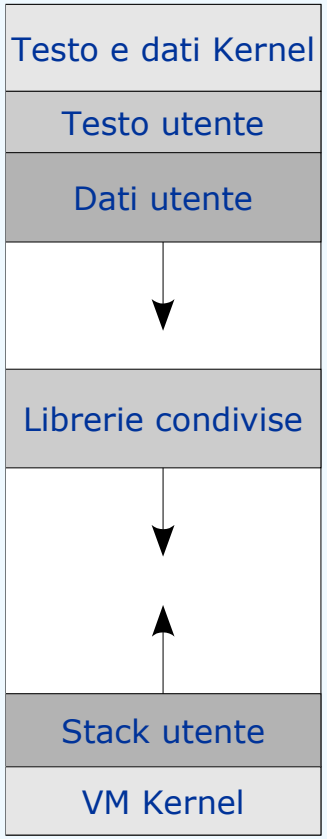
Nota storica: l'Internet Worm

- Come faceva il worm a scoprire nuovi host da infettare?
- Lettura molto interessante:
  - "The Internet Worm Program: An Analysis" by E.H. Spafford (http://www.textfiles.com/100/tr823.txt)
- Riflessioni:
  - Perchè il worm si è propagato così in fretta?
  - Un nuovo Internet worm oggi?
  - Eterogeneità, la struttura di Internet
  - Il problema delle gaming console

Buffer overflow

- È uno dei tipi di attacco più diffusi
- Linguaggi come C permettono di modificare i dati del programma in modo non coerente rispetto alla loro definizione
- Ad esempio: un array di 10 caratteri può essere riempito con 15 caratteri
- I byte in eccesso possono sovrascrivere zone **eseguibili** della memoria utente
- Inviando dati artefatti a un programma lo si può forzare ad eseguire codice arbitrario
- Per esempio, una shell!

Layout di memoria Linux



Buffer overflow

- Esempio banale di vulnerabilità locale: un programma inserisce in un array una stringa data senza controllarne la lunghezza
- Invocando il programma con una stringa opportuna lo si costringe a eseguire codice arbitrario (qui viene avviata una shell)
- In questo caso non ci sono conseguenze negative, la vulnerabilità è locale e il programma non ha permessi di superutente. Ma...
  - Cosa succede se si tratta di un server in ascolto sulla rete? Si riesce ad avere una **shell locale**!
  - Cosa succede se il programma ha permessi di superutente? Si riesce ad avere una **shell di root**!

Buffer overflow

- Si capisce perché è importante eseguire i programmi server con i privilegi minimi (soprattutto evitare root!)
- Nella maggior parte dei casi l'unico modo per evitare il problema e` non commettere questo genere di errori quando si scrive il programma
- Spesso si tratta di sviste banali e molto semplici da correggere (ad esempio controllare la lunghezza dei dati prima di copiarli altrove)
- Tuttavia, software molto famosi (server web e mail, client di posta, ...) hanno avuto storie molto tormentate in questo senso
  - Sendmail è famigerato per questo: ha contribuito fra l'altro alla diffusione dell'Internet Worm. È **troppo complicato**.

In pratica...

Vediamo nella pratica...
un buffer overflow “didattico”

Password brute-forcing

- I sistemi Unix mantengono le password degli utenti sotto forma di hash unidirezionale (facile da calcolare, difficile tornare indietro)
- Avendo a disposizione il file delle password si possono verificare molto velocemente tentativi "alla cieca"
- Quasi ovunque le password sono memorizzate in un file non leggibile, ma non sempre è così
- Esistono tool per cercare password banali o ottenibili in funzione di altri dati (ad esempio il nome dell'utente)
- Perché non usarli per controllare la qualità delle password dei **propri** utenti?

Password brute-forcing

- Uno dei più conosciuti è **John the Ripper**
- Usandolo ci si rende conto di quanto possano essere facili da trovare password ritenute “sicure”
- Allo stesso tempo si nota che è importante usare algoritmi di hashing complessi (come MD5), che rallentano l'attacco anche di un fattore 100
- Puo` essere configurato per inviare avvisi agli utenti segnalando la debolezza delle proprie password e invitandoli a cambiarle entro pochi giorni

In pratica...

Vediamo nella pratica...

John the Ripper

Vulnerabilità del kernel

- Anche il kernel può essere vulnerabile (anche nei moduli. Sicurezza dei moduli chiusi forniti da terze parti?)
- Esempi:
  - Stack TCP/IP, frammentazione (da **remoto**)
  - Memory management (da **locale**)
- Le vulnerabilità locali vengono da molti considerate poco pericolose ma questa è spesso un'errata valutazione del rischio. Esempio:

vulnerabilità remota (es. www/ftp/mail) ->
accesso **non privilegiato** ->
vulnerabilità locale ->
accesso con **privilegi di root**

Vulnerabilità del kernel: un esempio

- Linux kernel do_mremap VMA limit local privilege escalation vulnerability
  - Versioni vulnerabili: 2.2 fino a 2.2.25, 2.4 fino a 2.4.24, 2.6 fino a 2.6.2
  - http://www.isec.pl/vulnerabilities/isec-0014-mremap-unmap.txt
- La vulnerabilità riguarda una system call non privilegiata, mremap()
- È assente il controllo sul valore di ritorno della funzione do_mremap(), che in condizioni particolari può fallire. Sfruttando alcune condizioni particolari (attraverso un eseguibile "set user id") può portare all'esecuzione di codice con privilegi di root

Rootkit

- Set di tool utilizzato da un intrusore dopo aver violato un sistema. L'obiettivo è quello di garantirsi l'accesso e l'utilizzo mascherando la propria presenza e le tracce lasciate nel sistema

- Normalmente un rootkit si occupa di cancellare selettivamente parte dei log e di sostituire alcune utility tipiche con versioni apparentemente uguali ma "addomesticate" (es. opera sui login, processi e log)

- Esistono anche i "kernel rootkit" che lavorano a livello di system call. Vengono messi in opera attraverso moduli del kernel (nuovi o infettano quelli presenti) oppure modificando direttamente la memoria usata dal kernel (/dev/kmem). Le normali tecniche di rilevazioni per rootkit in questo caso non sono efficaci!

Rootkit

- Nonostante i rootkit cerchino di nascondere la loro esistenza, è molto difficile rendersi completamente invisibili
- Ci sono alcuni tool che cercano di capire se il sistema è stato compromesso da rootkit noti. Ad esempio, **chkrootkit**.
- I rootkit noti sono abbastanza facili da rilevare; per gli altri esiste solitamente una serie di comportamenti tipici che possono "tradirli"
- Spesso i rootkit cercano di nascondere l'esistenza di alcuni processi, modificando i binari di alcuni programmi (ps, top) o interagendo con /proc
- Questo porta a incoerenze rilevabili
- Vediamo un esempio di rootkit rilevato tramite chkrootkit

In pratica...

Vediamo nella pratica...
ARK vs chkrootkit

Filesystem-based Intrusion Detection System (IDS) - AIDE

Advanced Intrusion Detection Environment (AIDE)

- È basato su un database che contiene una serie di informazioni su ogni singolo file e directory che fanno parte del filesystem
- Quale parte del filesystem considerare è esprimibile attraverso espressioni regolari
- Lo scopo è quello di verificare l'integrità dei singoli file, rilevando e riportando eventuali modifiche:
  - ora e data di creazione e modifica
  - dimensione
  - algoritmi di digest (md5, sha1, rmd160, tiger, haval ecc)
- Il database generato **non è criptato e nemmeno firmato**

In pratica...

Vediamo nella pratica...
aide

Filesystem-based IDS - Tripwire

- È basato su un database che contiene una serie di informazioni su ogni singolo file e directory che fanno parte del filesystem
- Esiste in varie versioni commerciali e libere
- Nel funzionamento è estremamente simile ad aide
- Supporta firma e crittografia del database generato attravero le firme dei file analizzati

- Alla luce di quanto visto: **Tripwire è più sicuro di AIDE?**
- È sufficiente garantire una buona integrità del database per essere sicuri che il sistema è integro?

In pratica...

Vediamo nella pratica...
tripwire

Filesystem-based IDS - Samhain

- Samhain è un tool simile ad Aide e Tripwire ma che integra il supporto per un repository centralizzato (struttura client-server)
- È basato come i precedenti su checksum crittografici
- Il server è basato su un DBMS (es. PostgreSQL, MySql ecc)
- I client (samhain) si collegano al server (yule) attraverso connessioni crittografate
- Configurazione e database possono essere firmati con OpenPGP, la crittografia è basata su AES
- I client si "autenticano" al server attraverso una "embedded key" per migliorare la resistenza ai trojan

Linux: alla ricerca della sicurezza

Esistono delle versioni particolari di Linux sviluppate con il preciso obiettivo di migliorare la sicurezza

- Alcuni esempi:
  - Security Enchanced Linux (http://selinux.sourceforge.net/)
  - Adamantix (http://www.adamantix.org/)
  - Trustix (http://www.trustix.net/)
- Le modifiche principalmente riguardano:
  - **Modelli per l'Access Control**: Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role-Based Access Control (RBAC) + Multi Level Security Model ("Top Secret", "Secret", "Confidential"...)
  - **Protezione dai buffer overflow**. Ad esempio PaX (Kernel), SPP (gcc patch)
  - **Crittografia** del filesystem

Port scanner

- Uno dei primi passi compiuti da chi vuole violare un sistema dall'esterno è cercare di scoprirne sistema operativo usato e servizi attivi
- In base a questi dati si possono poi cercare debolezze specifiche
- Inoltre, si possono scoprire porte rimaste aperte per errori o leggerezze dell'amministratore (esempio: il compilatore distribuito **distcc**)
- I portscan per porte specifiche possono essere molto veloci e scandire intere sottoreti in breve tempo
- Vale la pena fare un portscan verso se' stessi per vedere come si appare dall'esterno

Port scanner

- Uno degli strumenti più noti è **nmap**
- La sua funzione principale è controllare quali porte sono aperte su un host o una sottorete intera di macchine
- Può farlo in vari modi, fra cui alcuni che cercano di "non farsi notare troppo" da chi subisce il portscan
- Inoltre può cercare di capire che sistema operativo e che versione è in uso analizzando, per esempio, il modo in cui vengono generati i numeri di sequenza TCP (OS fingerprinting)
- Di default non vengono provate tutte le 65535 porte (ci vuole tempo!)
- Usare porte non standard può avere senso?

In pratica...

Vediamo nella pratica...
nmap

Security scanner

- Esistono tool che non si limitano a controllare che una porta sia aperta o chiusa
- Contengono database di vulnerabilità note e per ogni servizio trovato cercano di capire quali di queste siano presenti sul sistema in esame
- Possono essere molto utili come semplice controllo "automatico" da fare per accertarsi di non essere vulnerabili agli attacchi più comuni

- **Satan** è uno dei "venerabili", è ancora utile perchè prodotti più nuovi tendono a non controllare le vulnerabilità poco recenti

- **Nessus** ne è un derivato, più moderno e attivamente mantenuto

In pratica...

Vediamo nella pratica...

nessus

Firewall

- Un buon firewall è essenziale per qualsiasi politica di sicurezza in rete (dal modem alla LAN più complessa)
- Solitamente è necessario sia un sistema di filtri a livello IP (**netfilter**) che una serie di **proxy** a livello applicazione
- Lo scopo di un firewall non è solo quello più ovvio:
  - **Tagliare fuori** connessioni indesiderate
  - Impedire l'**invio** di dati sensibili **all'esterno**
  - Proteggersi dagli **attacchi interni**
- Una regola molto importante: tutto ciò che non è esplicitamente permesso è vietato
- Vediamo un esempio di netfilter: Linux e iptables

Firewall: iptables

- Iptables ha avuto diversi predecessori: ipfwadm fino a 2.0, ipchains fino a 2.2 e iptables da 2.4 in poi
- Ogni pacchetto attraversa una serie di **catene** di regole, ognuna delle quali specifica ad es. indirizzo e porta di provenienza o destinazione (ci sono anche regole molto piu' sofisticate)
- Una novità di iptables è di essere un filtro **stateful**: i pacchetti sono esaminati nel contesto della connessione cui appartengono
- I log ottenibili possono essere analizzati (anche automaticamente) per rendersi conto di “chi” sta cercando di fare “cosa” sulla rete
- In alternativa ai log, alcuni tool (es. **snort**) analizzano il traffico e raccolgono informazioni dettagliate sul traffico anomalo

In pratica...

Vediamo nella pratica...
iptables

Note

- Nella realizzazione di queste slide non è stato fatto alcun male a **server reali** con bit reali: solamente **server e reti virtuali**, si prega vivamente di fare altrettanto!
- La sicurezza è molto divertente (almeno fino a quando non si viene bucati) ma pone di fronte anche a vari problemi etici. Spesso è il caso di riflettere anche su questi e non solamente sulla parte prettamente tecnica
- Molte intrusioni non vengono rilevate se non dopo molto tempo o eventualmente mai. Non sei mai stato bucato o non ti sei mai accorto di esserlo stato?

- I tool posso aiutare molto ma a fare la differenza sono comunque la preparazione e le capacità

Note e bibliografia

- "The Internet Worm Program: An Analysis". E.H. Spafford. http://www.textfiles.com/100/tr823.txt
- "Practical Unix and Internet Security". S. Garfinkel, G. Spafford
- "Building Internet Firewalls". E.D. Zwicky, S. Cooper, D.B. Chapman
- "Cuckoo's Egg". C. Stoll
- SELinux. K. Thomson. http://www.samag.com/documents/s=7835/sam0303a/0303a.htm http://www.nsa.gov/selinux/
- Nessus. http://www.nessus.org/
- Nmap. www.insecure.org/nmap/
- Aide. www.cs.tut.fi/~rammer/aide.html
- Tripwire. http://www.tripwire.com/
- BugTraq. http://www.securityfocus.com/archive/1
- SecurityFocus. http://www.securityfocus.com/
- Open Source Vulnerability Database. http://www.osvdb.org/
- Debian Security Information. http://www.debian.org/security/