Introduzione alla Sicurezza dei Sistemi

(prima parte)

Gabriele D'Angelo

gda@cs.unibo.it

http://www.cs.unibo.it/~gdangelo/

Università di Bologna

Corso di Laurea in Scienze dell'Informazione

Novembre, 2006

Scaletta della lezione

- Sicurezza dei sistemi: concetti di base
- Autenticazione, Identificazione
- Problematiche relative alle password
- Denial of Service (DoS)
- Virus e Worm
- Il concetto di vulnerabilità
- L'Internet Worm (Morris Worm)
- Introduzione ai buffer overflow
- Cavalli di troia
- Security by obscurity
- Social engineering
- Note e bibliografia

Warning!

- Quanto vi presenterò oggi e nelle prossime lezioni è a

  puro scopo didattico

- Le intrusioni nei sistemi informatici sono punite dalla legge, *sia civilmente che penalmente*, quindi non è il caso di giocare senza capire in modo approfondito cosa si sta facendo e quali sono le conseguenze delle proprie azioni

- Esistono macchine e reti **virtuali**: sono ottime per sperimentare!

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

Sicurezza dei sistemi: una o molte definizioni?

- Cosa intendiamo quando parliamo di **sicurezza informatica** o più specificatamente di **sicurezza dei sistemi**?

- Una lista molto incompleta di interpretazioni:
  - **Identificazione – Autenticazione - Autorizzazione**
  - **Disponibilità – Affidabilità**
  - **Integrità**
  - **Confidenzialità – Privacy**
  - **Anonimato**

Autenticazione

La base di ogni sistema di sicurezza è solitamente centrata su **autenticazione / identificazione**

- Meccanismi più diffusi:
  - qualcosa che l'utente **conosce** (es. **password** o in generale un segreto)
  - qualcosa che l'utente **ha** (es. **oggetto fisico**)
  - qualcosa che l'utente **è** (es. meccanismi **biometrici**)
- Vantaggi e svantaggi dei vari meccanismi

Non c'è nulla da proteggere?

- **Cosa** vogliamo proteggere e **da cosa** vogliamo proteggerci?
- Obiettivi:
  - **Confidenzialità** dei dati vs. Esposizione dei dati
  - **Integrità** dei dati vs. Manomissione dei dati
  - **Disponibilità** del sistema vs. Denial of Service (DoS)
- In alcuni casi vogliamo proteggere il nostro nome e la nostra **rispettabilità** (ad esempio non incorrere in problemi legali)
- Non vogliamo che vengano diffusi **dati sensibili** (informazioni mediche, religiose, politiche ecc.) e non vogliamo che i nostri dati vengano **modificati** senza autorizzazione
- In altri casi vogliamo semplicemente avere il sistema a **disposizione** quando ne abbiamo bisogno!

Sicurezza: host, home, Internet

- Molti dei **principi fondamentali** della sicurezza dei sistemi sono validi indipendentemente dallo scenario di applicazione

- Ma i vari scenari sono molto diversi, con problematiche tipiche:
  - **Host** security
  - **Home network** security
  - **Internet** security

- Nell'ambito di queste lezioni, tranne poche eccezioni, cercheremo di concentrarci solamente sui principi di base e sulla host security

- Affrontare in modo serio i problemi di Internet security richiede, tra l'altro, una conoscenza approfondita delle architetture di rete e dei protocolli

IDENTIFICAZIONE : meccanismi basati su password

La maggior parte degli *utenti* sono:

- portati a scegliere **password banali**, basate su parole di dizionario oppure su dati personali pubblici, preferenze ed interessi facilmente individuabili
- molto restii a **cambiare** password
- abituati a scrivere le password in luoghi insicuri

- Le password sicure sono difficili da ricordare
- Moltiplicazione delle password: la stessa password viene utilizzata per servizi differenti
- **Principio generale**: *la sicurezza di un sistema è stimabile come la sicurezza del suo anello più debole*
- Le password sono spesso molto facili da **sniffare** (es. protocolli di comunicazione e file di salvataggio in chiaro)

Password brute-forcing: “procedere per tentativi”

- Nei sistemi Unix (così come molti altri) le password degli utenti vengono mantenute sotto forma di **hash unidirezionale** (ovvero una funzione che sia facile da calcolare ma difficile da invertire)
- Avendo a disposizione il file delle password si possono verificare molto velocemente tentativi “alla cieca”, *in modalità offline*
- Quasi ovunque (nei sistemi moderni) le password sono memorizzate in un file non leggibile da tutti, *ma non sempre è così*
- Esistono degli strumenti per cercare password banali o ottenibili in funzione di altri dati (ad esempio il nome dell'utente)
- Questi strumenti hanno un uso offensivo ma anche (e soprattutto) difensivo: possono essere usati per controllare periodicamente la *qualità delle password* dei **propri** utenti

Password brute-forcing: John the Ripper

- Uno dei più famosi e diffusi è **John the Ripper**
- Usandolo ci si rende conto di quanto siano facili da trovare password ritenute “sicure”
- Allo stesso tempo si nota l’importanza di utilizzare algoritmi di hashing complessi (come MD5). Il loro costo computazionale ha l’effetto di rallentare l’attacco e quindi renderlo meno praticabile
- Uso “legittimo”: usare questo strumento per inviare avvisi agli utenti segnalando la debolezza delle proprie password e invitandoli a cambiarle entro pochi giorni. Quello che si fa è *prevenire un attacco emulandolo a priori*

DISPONIBILITÀ: Denial of Service (DoS)

- **Denial of Service (DoS)** in questo caso l'obiettivo è l'*interruzione* o il *rallentamento* di un servizio
- Le motivazioni possono essere molteplici e spesso non immediatamente chiare. Ad esempio abbattere un server per *prenderne illegittimamente il posto* attraverso una macchina costruita appositamente per impersonarlo
- Gli attacchi di tipo DoS sono **estremamente diffusi** e spesso realizzati in modo tecnicamente grossolano: possono richiedere (non sempre) un livello di conoscenze piuttosto limitato

- **Fork bomb**: cosa succede lanciando un processo che si moltiplica ricorsivamente all'infinito? *Quali privilegi sono richiesti all'utente per attuare un attacco di questo genere?*

Virus e Worm

- Spesso si parla indistintamente di Virus e di Worm, creando molta confusione, nascono come concetti differenti:
  - **Virus**: programma o parte di programma che, mimando i virus biologici, infetta altri eseguibili (usualmente all'interno dello stesso host, ultimamente il termine si applica anche per le infezioni tra computer diversi ma collegati in rete). Sono degni di nota anche i macro-virus
  - **Worm**: programma in grado di replicarsi che, sfruttando delle *vulnerabilità dei sistemi* coinvolti, tenta di diffondersi infettando vari host collegati in rete
- *Per colpa di un worm è possibile che Internet collassi?*

Nota storica: l'Internet Worm

Lasciamo solo per un attimo la host security per analizzare uno dei più famosi e citati incidenti di sicurezza: l'**Internet Worm** detto anche **Morris Worm** dal nome del suo creatore

- **Vulnerabilità sfruttate dal worm**:
  - *Buffer overflow* in fingerd
  - Il mail server sendmail (attraverso il codice di DEBUG)
  - Remote shell 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 sono riusciti ad impedire infezioni multiple dello stesso server -> **DoS**

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 velocemente?
  - Sarebbe possibile un nuovo Internet worm oggi?
  - Eterogeneità, la struttura di Internet, sistemi operativi?
  - Il problema delle gaming console

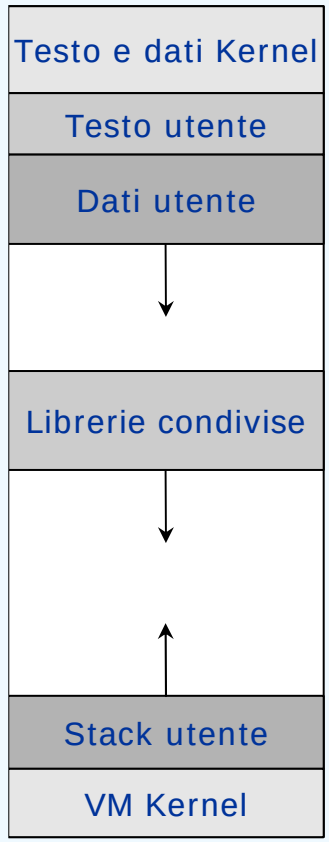
Il concetto di vulnerabilità

- Il concetto di **vulnerabilità:**
  - **una debolezza di un sistema che permetta ad un attaccante di violarne l'integrità, la confidenzialità, il controllo degli accessi…**
    - Il software non è perfetto: *gli errori sono estremamente comuni*. Errori banali possono avere conseguenze del tutto impreviste ed *estremamente difficili da valutare a priori*
    - Facendo leva su alcuni errori è possibile **compromettere** un software e in alcuni casi **la sicurezza di tutto il sistema**
    - Il problema dei **buffer overflow**: molte vulnerabilità derivano da errori di progettazione o da **banali errori di programmazione**
    - L'errore più diffuso è un'errata **gestione della memoria e della gestione dell'input**
    - Si definisce **exploit** *un software in grado di sfruttare una vulnerabilità*

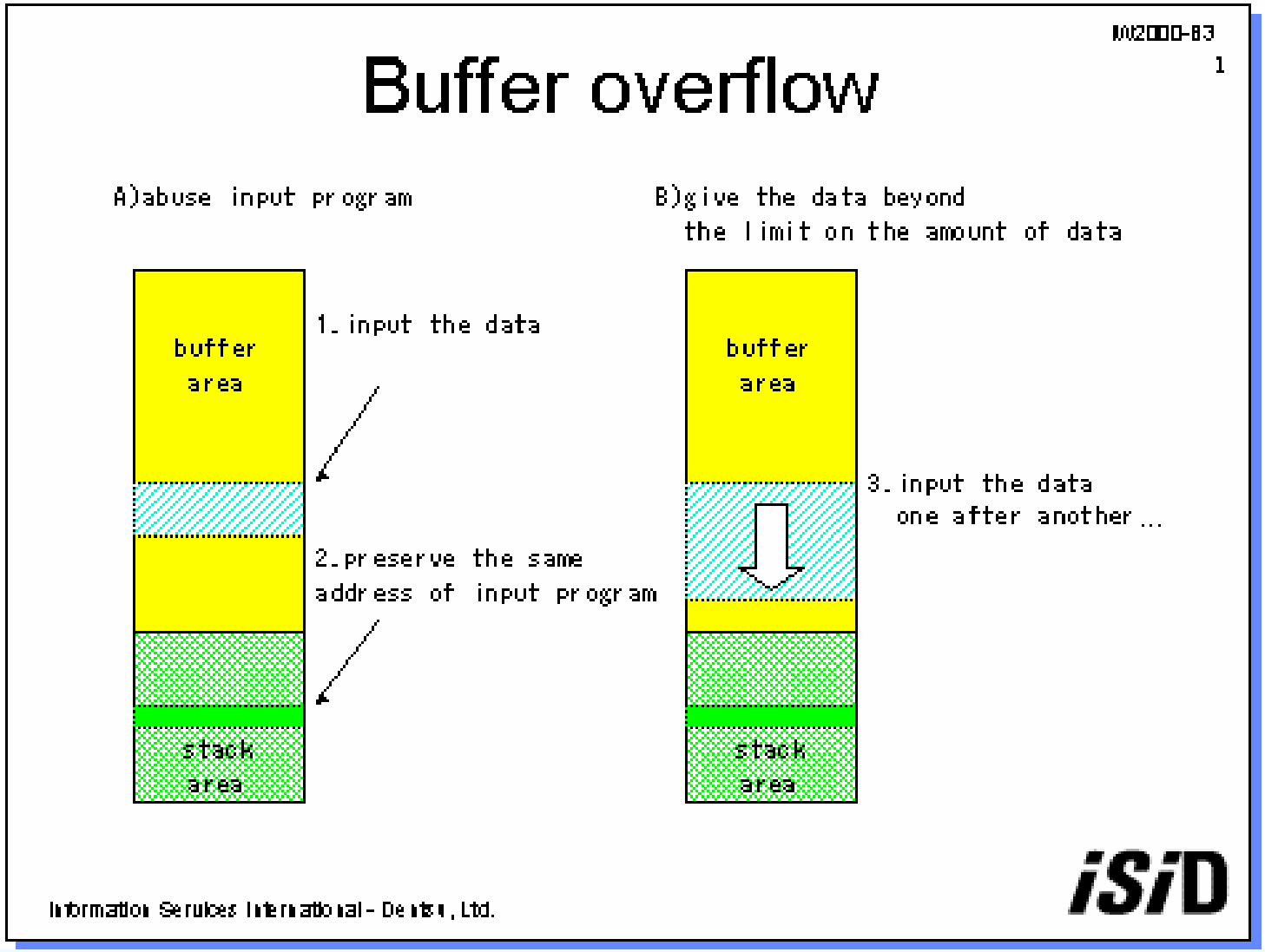
Buffer overflow (1/3)

- È di gran lunga la vulnerabilità **più diffusa**
- 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 (2/3)



Buffer overflow (3/3)