Sicurezze e False Sicurezze

Un confronto aperto tra modelli “free” e “proprietari”

Stefano Zanero

Dipartimento di Elettronica e Informazione
Politecnico di Milano
zanero@elet.polimi.it

LinuxDay ’04, 27/11/2004

L'esigenza della sicurezza

- Un tempo i sistemi "sicuri" erano pochi, oscuri, acceduti infrequentemente.
- Oggi, quasi tutti i sistemi contengono dati importanti, critici o semplicemente privati, e vanno protetti. Al contempo, devono essere sempre più aperti e interconnessi.
- "Virtual is real... open is secure": oltre che un ottimo motto pubblicitario, una verità
- Esigiamo una "garanzia di sicurezza" dai sistemi informatici

Cos'è la sicurezza ?

- Definiamo accademicamente la sicurezza come il raggiungimento di 3 obiettivi:
  - CONFIDENZIALITÀ
  - INTEGRITÀ
  - DISPONIBILITÀ
- Accesso, in lettura o in scrittura, "if and only if" l'utente è autorizzato appropriatamente.
- Tutto ciò è di una semplicità ingannevole

Molti punti di vulnerabilità ...

- I sistemi informativi devono essere progettati per garantire la sicurezza, mediante un uso appropriato e ragionevole di tecniche di criptografia, autenticazione, auditing, controllo.
- I processi devono essere pensati per la sicurezza: per esempio, l'assegnazione di login e password
- Le persone sono vulnerabili: possono essere convinte o costrette a utilizzare male i loro privilegi; possono essere risentite e volersi vendicare
- **Il software deve essere sicuro e robusto, comportarsi nel modo in cui è previsto che si comporti, sempre**

Sicurezza del software ?

Non esistono software sicuri a priori!

- Ogni progetto software può contenere vari tipi di errori e problemi:
- INTERNI
  - Di tipo concettuale (errori di design)
  - Di tipo programmativo
- ESTERNI
  - Causati da librerie e componenti riutilizzati
  - Causati dall'interazione con altri programmi e con il sistema operativo
  - Causati da misconfigurazione

Software libero vs. Software proprietario

Software Libero:

- Codice sorgente disponibile a chiunque
- Programmato e concepito da appassionati
- Liberamente modificabile e ridistribuibile
- Non sottoposto ad esigenze economiche e di marketing tipiche di un vendor

Software Proprietario:

- Nesssuno ha accesso al codice sorgente a parte gli sviluppatori
- Sviluppato da programmatori di professione
- Non modificabile, se non con difficoltà (reverse engineering); non ridistribuibile
- Sostenuto dai capitali investiti da società interessate al ROI

I riflessi sulla sicurezza

Software Libero:

- Il sorgente può essere auditato da chiunque
- Patch, e sviluppi alternativi possono essere creati liberamente
- Collaborazione e riutilizzo di codice tra progetti
- Rapidissimo processo di code/audit/exploit/patch

Software Proprietario:

- Security through obscurity
- È difficile fare auditing, ricerca dei bug black box e con disassemblamento
- "Reinventing the wheel"
- Anche trovando un bug, è difficile comprenderne esattamente la natura, e spesso i vendor non approfondiscono

Problemi simmetrici in entrambi i mondi

Software Libero:

- Non sai "chi è il padre" del codice che usi (trojans, anyone?)
- Non ci sono impegni commerciali: le release possono avvenire troppo presto... o mai.
- Falso senso di "sicurezza a priori" per la presenza del codice... ma l'avete letto ?

Software Proprietario:

- Rassegnata fiducia in quello che state eseguendo (spyware and backdoors, anyone ?)
- Dipendenza totale dalla casa produttrice per il rilascio di patch . . . e a volte bisogna convincerli!
- Il fatto che non ci siano bug pubblici non significa che non ne esistano . . .

Non viviamo nel migliore dei mondi possibili . . .

- La morale è che non è corretto a priori affermare che il software libe sia “più sicuro” di quello proprietario.
- Volete un esempio ? Andate su: `http://packetstormsecurity.nl` e troverete exploit in abbondanza per ogni piattaforma che vi venga in mente . . .
- Addirittura, ad un’occhiata superficiale, vi sono più exploit per programmi open source!

Un problema anche economico!

- Noi sostenitori dell'open source siamo abituati a discorsi romantici, ma la sicurezza ha un *solido conto* da presentare
- Gli investimenti per aggiornare e riscrivere il codice non sicuro sono notevoli, per gli stessi produttori !
- Lo sforzo economico e lavorativo per mantenere "up to date" l'ambiente di produzione è enorme
- Il fattore rischio creato dall'insicurezza non va trascurato: la diminuzione del rischio operativo è tradizionalmente considerata un obbiettivo per qualsiasi azienda !
- E non abbiamo nemmeno parlato dei *danni* ...

Full Disclosure

- Strettamente connesso al discorso dell'open source è il tema della *full disclosure*, che potrebbe essere visto come *un metodo "open" di analisi del software*
- Full disclosure significa la pratica, diffusasi negli ultimi anni, di annunciare pubblicamente le vulnerabilità scoperte, diffondendo spesso anche degli exploit dimostrativi.
- Molti esperti di sicurezza applicano la full disclosure su forum aperti (come bugtraq, `http://online.securityfocus.com`), in genere dopo aver avvertito del problema i produttori ed aver loro concesso un lasso di tempo ragionevole per risolverlo.

Non è tutto oro . . .

- La full disclosure, specie su progetti open source in cui la advisory spesso contiene anche una patch preliminare, ha contribuito alla soluzione di molti problemi di sicurezza
- La full disclosure ha contribuito a rendere il pubblico conscio delle problematiche di sicurezza (awareness)
- Tuttavia, molti indicano la disclosure come responsabile dell'aumento vertiginoso del fenomeno degli "script kiddies"
- Il fatto che quelli che si lamentano siano in gran parte esponenti del mondo dei vendor di software proprietario pone qualche legittimo dubbio. . .
- I danni causati dai ragazzini hanno suscitato un fiorente mercato per le società di sicurezza informatica, su alcune delle quali si potrebbero affrontare lunghi discorsi...
- In alcuni casi si è assistito a fenomeni di *repressione* della libertà di ricerca e di diffusione delle advisory

Security, “free” style !

- Anche nel campo dei prodotti per la creazione di sistemi di sicurezza esistono progetti free di qualità pari, se non superiore, agli equivalenti del mondo commerciale:
- Firewall: IPFilter, IPChains...
- IDS: Snort, LIDS, AIDE, TripWire...
- Crittografia: GPG
- Implementazioni di IPSEC, Kerberos
- Vulnerability scanner: Nessus
- Antivirus: CLAMAV (c’è ancora della strada da fare, ma è un ottimo inizio!)
- Metodologie: Open Source Security Testing Methodology Manual

Conclusioni

- Free *non* significa intrinsecamente più sicuro
- Free *non* significa genericamente fatto meglio
- Scegliere il software libero lascia all'utente e all'amministratore di sistema la possibilità di adoperarsi *attivamente* per proteggere la propria sicurezza.
- Scegliere software proprietario significa affidarsi passivamente alle capacità e alla buona volontà (capirai...) dei vendor.
- In conclusione, il modello open source responsabilizza la comunità e l'utente, e li mette in grado di agire, ma *non* è la panacea di tutti i mali.

Link utili

- The Open Source Testing Methodology Manual (OSSTMM): `http://www.isecom.org`
- Mailing list italiane sulla sicurezza informatica: `http://www.sikurezza.org`
- SecurityFocus, portale sulla sicurezza: `http://online.securityfocus.com`
- RFPolicy, una policy per la full disclosure: `http://www.wiretrip.net/rfp/policy.html`
- Copie delle slide, altre presentazioni e articoli scientifici: `http://www.elet.polimi.it/upload/zanero`