GM Servizi

Assessment di sicurezza

(metodologia seguita)

Hacking Team S.r.l.	http://www.hackingteam.it
Via della Moscova, 13 *20121 MILANO (MI) - Italy*	info@hackingteam.it
Tel. +39.02.29060603	*Fax +39.02.63118946*

STORIA DEL DOCUMENTO		
Versione	Data	Modifiche Effettuate
1.0	21 Dicembre 2007	Emissione

INFORMAZIONI	
Data di Emissione	21 Dicembre 2007
Versione	1.0
Tipologia Documento	Documento metodologico
Numero di Protocollo	//
Numero Pagine	18
Redatto da	
Verificato da	
Approvato da	

INDICE

1 L'approccio utilizzato

La metodologia di *assessment* si fonda su un approccio iterativo, che prevede l'esecuzione ciclica di cinque attività:

- Analisi
- Identificazione delle vulnerabilità
- Definizione degli scenari di attacco
- Esecuzione degli attacchi/validazione degli scenari
- Definizione delle contromisure

La necessità di operare in modo iterativo nasce dal fatto che l'esecuzione di ogni attacco può accrescere il livello di conoscenza del sistema e/o il livello di privilegio di chi lo esegue, rendendo necessaria una nuova fase di analisi, sulla cui base identificare nuovi scenari di attacco.

L'*assessment* si conclude quando, sulla base di tutte le informazioni raccolte e dei privilegi ottenuti, non si possono identificare ulteriori scenari di attacco.

L'*output* dell'*assessment* è costituito dal presente documento, che descrive:

- come la metodologia è stata applicata al caso particolare in esame
- i risultati degli attacchi effettuati
- le contromisure da adottare per eliminare le vulnerabilità individuate

I prossimi paragrafi descrivono in maggior dettaglio le modalità di esecuzione delle singole attività previste dalla metodologia utilizzata.

1.1 Analisi

Questa fase consiste nella raccolta di informazioni relative ai sistemi oggetto del *vulnerability assessment*, sulla cui base definire la strategia di analisi e di attacco.

Le informazioni raccolte sono di carattere funzionale (scopo del sistema, tipologie di utenti, dati trattati…) e di tipo tecnico (piattaforma *HW/SW*, tecnologie, servizi presenti…)

Le modalità di reperimento delle informazioni sono molteplici:

- incontri con sviluppatori e/o responsabili
- documentazione fornita dal Cliente
- analisi diretta del sistema

1.2 Identificazione delle vulnerabilità

Questa fase consiste nella ricerca di errori logico/architetturali ed implementativi che possono essere sfruttati per compromettere la sicurezza del sistema.

Le tecniche utilizzate per le due classi di errori sono differenti.

- Gli ***errori di tipo logico/architetturale*** dipendono dall'applicazione in esame; la loro identificazione deve essere fatta manualmente, da parte di figure professionali che, sulla base della propria esperienza, siano in grado di riconoscere *pattern* di vulnerabilità generici nella logica di una particolare applicazione. In alcuni casi, l'analisi manuale può portare ad identificare anche debolezze architetturali specifiche, non riconducibili a nessun *pattern* noto. In questa fase, l'utilizzo di *tool* automatici è limitato ai *proxy* HTTP, che permettono di analizzare e modificare i flussi *input/output* dell'applicazione
- Gli ***errori implementativi*** vengono identificati analizzando l'*output* prodotto dall'applicazione in risposta ad *input* costruiti in modo tale da produrre comportamenti inattesi. Utilizzando un *database* contenente vulnerabilità note e relative tecniche di attacco, si procede mediante applicazione sistematica di tali tecniche di attacco, analizzando poi l'*output* ottenuto. La ricerca di errori implementativi può essere efficacemente supportata da *tool* automatici, che garantiscono velocità e completezza, sia in termini di parametri di *input* analizzati, sia in termini di *test* effettuati. L'intervento umano rimane comunque necessario per l'eliminazione dei falsi positivi, cioè quelle vulnerabilità individuate dai *tool* automatici, che non sono effettivamente presenti nell'applicazione

1.3 Definizione degli scenari di attacco

L'individuazione di una singola vulnerabilità non è sufficiente, nella maggior parte dei casi, a compromettere la sicurezza di un sistema. Un attacco che abbia come obiettivo l'accesso ad informazioni riservate, il furto di identità ad altri utenti, o, in generale, qualsiasi utilizzo non consentito del sistema richiede l'elaborazione di una strategia articolata che comprende relative tecniche e sfrutta diverse vulnerabilità.

È quindi necessario, sulla base dei risultati della fase di individuazione delle vulnerabilità, definire le strategie di attacco che possono essere utilizzate per uno specifico sistema. Questa attività deve essere svolta caso per caso in modo completamente manuale, da parte di figure professionali con competenze sia di livello architetturale/sistemistica, sia implementativo.

1.4 Esecuzione degli attacchi

Poiché il *vulnerability assessment* viene svolto senza avere accesso al sistema, l'effetto di un attacco può essere determinato soltanto mediante la sua effettuazione. Questa fase può richiedere l'utilizzo di programmi *ad hoc* per realizzare gli attacchi individuati nella fase precedente.

1.5 Definizione delle contromisure

L'attività di *vulnerability assessment* si conclude con la stesura di un *report* che descrive l'analisi svolta, gli scenari di attacco individuati, i tentativi di attacco eseguiti ed il loro esito. Per gli attacchi che hanno avuto successo viene riportata la descrizione delle vulnerabilità che lo hanno reso possibile, gli errori logici o implementativi che ne sono la causa e gli interventi correttivi necessari per l'eliminazione del problema.

2 Analisi di rete e sistemistica

La maggior parte delle attività di analisi eseguite tentano di emulare il comportamento di un vero *hacker*.

2.1 Analisi non invasiva

FOOTPRINTING

Questa fase ha lo scopo di raccogliere il maggior numero di informazioni sull'obiettivo che si intende attaccare senza impattare l'obiettivo stesso, ovvero effettuando una cosiddetta analisi non invasiva. In particolare in questa fase si cerca di determinare *domini, blocchi di rete, e indirizzi IP dei sistemi direttamente collegati a Internet*. Gli strumenti utilizzati sono Search Engine, Whois server, Arin database, DNS…

SCANNING

L'obiettivo dello *scanning* è ottenere una mappa più dettagliata possibile del sistema da attaccare. Ciò significa acquisire informazioni sugli IP dei blocchi di rete trovati nella fase precedente (*IP discovery*), sui servizi attivi (*TCP/UDP port scan*) e, infine, sui sistemi operativi. Gli strumenti utilizzati sono interrogazioni ICMP (gping, fping…), scansione delle porte TCP e UDP (strobe, netcat, nmap…), *fingerprint dello stack* (nmap, ethercap).

2.2 Analisi invasiva

ENUMERATION

In questa fase si effettuano connessioni dirette ai *server* e interrogazioni" esplicite. Tali attività potrebbero, a seconda della configurazione presente sui sistemi *target*, originare dei *logs* sui sistemi (tipicamente su sistemi di controllo). Attraverso l'enumerazione si vuole giungere a identificare, sulle macchine riscontrate come raggiungibili, *account* validi (*list user accounts*), risorse condivise (*list file shares*) e applicazioni attive sulle porte in ascolto (*identify application*). Le tecniche utilizzate variano a seconda dei sistemi operativi delle macchine che vogliamo analizzare.

2.3 Attacco

GAINING ACCESS

Una volta ottenute le informazioni del punto precedente, i tester tentano di "entrare" nel sistema remoto. I metodi utilizzati anche in questo caso dipendono dal sistema operativo della macchina *target*, ma si basano sostanzialmente sulla ricerca di *password* corrispondenti agli utenti trovati (*password guessing*), sullo sfruttamento di errori progettuali delle applicazioni e servizi attivi sul *server* (*buffer overflows*, attacchi *data driven*, ecc.) o del sistema operativo stesso.

ESCALATING PRIVILEGES

L'obiettivo di questa fase è sfruttare i risultati ottenuti nella fase precedente per ottenere il pieno controllo del sistema remoto attaccato. Ciò si ottiene, per esempio, reperendo i *files* presenti sul sistema che contengono le *password* (/etc/passwd, SAM) e tentando di decifrare le *password* in essi contenute (*password cracking*), oppure utilizzando appositi *exploits*.

2.4 Consolidamento

PILFERING

Se si giunge a questa fase significa che si è ottenuto il pieno controllo del sistema *target*. Quindi è bene valutare la configurazione del sistema stesso al fine di capire se, dove e cosa il sistema registra (*logs*). I sistemi di *auditing* saranno eventualmente disabilitati (es. con Win NT mediante auditpol). A questo punto la macchina in oggetto può diventare una "testa di ponte" per attaccare altre macchine. In tal caso saranno reperite informazioni riguardanti altri sistemi.

COVERING TRACES AND CREATING BACK DOORS

Prima di abbandonare il sistema "conquistato" vengono cancellati gli eventuali *logs* che hanno registrato la presenza clandestina ed eventualmente installati *trojan* o *back-doors* che consentano di rientrare facilmente sulla macchina in un secondo momento. Può essere utile anche installare *tools* nascosti quali *sniffers* o *keyloggers* al fine di catturare altre *password* del sistema locale o di altri sistemi ai quali utenti ignari si collegano dalla macchina controllata.

3 Strumenti utilizzati

Nella fase di analisi ci si è avvalsi dei seguenti tools:

- ***Network mapping tool:*** *tool* che eseguono una scansione di singoli sistemi oppure intere reti al fine di determinare le porte aperte, le applicazioni che sono in ascolto in quelle porte, il tipo e la versione approssimata del sistema operativo…
- ***Known exploits***: spesso le vulnerabilità possono essere sfruttate utilizzando dei codici creati appositamente per ottenere l'accesso tramite il servizio vulnerabile presente sul sistema

4 Principi generali

I principi metodologici che seguono sono stati impiegati nella valutazione dei livelli di sicurezza e nella formulazione delle soluzioni tecniche proposte.

4.1 Principio del privilegio minimo

"Tutto quello che non è strettamente necessario deve essere eliminato"

È il principio più importante da seguire in materia di sicurezza. Il principio del privilegio minimo *minimum privilege* afferma che ogni *soggetto* all'interno di un sistema informatico (utenti, processi, programmi) deve essere in grado di accedere solamente agli *oggetti* del sistema (dati, accessi, flussi di dati, operazioni sui dati) di cui ha strettamente bisogno per le proprie funzioni. Il principio del privilegio minimo è fondamentale, perchè limita l'esposizione degli oggetti ad eventuali attacchi e, al tempo stesso, limita i danni subiti dall'intero sistema nel caso che un "attacco" abbia successo.

4.2 Principio della ridondanza

"Ogni meccanismo di sicurezza si può inceppare"

La sicurezza di un sistema (o di una procedura, di una funzione, di un'applicazione) non deve dipendere da un solo meccanismo di sicurezza, per quanto esso possa sembrare robusto e infallibile. È sempre auspicabile prevedere delle soluzioni di *backup* che possano intervenire nell'evenienza di una temporanea indisponibilità di una risorsa adibita alla protezione del sistema o in presenza di un "attacco" sferrato contro la risorsa stessa.

Per esempio, è buona norma duplicare le procedure di logging quando l'*auditing* delle applicazioni è *security-critical* per il business aziendale. Oppure, assumere che le misure di sicurezza principali per il

controllo dell'integrità possano in qualche modo essere scavalcate e impiegare dei sistemi di controllo di flusso che abbiano la funzionalità di controllare che le misure di sicurezza principali siano ben funzionanti.
A supporto di quanto è stato detto, bisogna osservare che tutte le tecnologie di *security* soffrono di un'obsolescenza assai più rapida rispetto agli strumenti software convenzionali.
La qualità e l'efficacia degli attacchi che possono essere effettuati contro un sistema informatico è in costante evoluzione, e per questa ragione è necessario che le misure di sicurezza rispecchino le nuove tecniche di attacco non appena queste diventano note.
Internet è un formidabile catalizzatore del processo evolutivo "nuovo attacco - nuova misura di sicurezza per rendere inefficace l'attacco - nuovo attacco in grado di neutralizzare la misura di sicurezza precedente".
È opportuno ipotizzare che anche il personale interno all'azienda possa essere in grado di procurarsi informazioni e tecnologie sufficienti a sfruttare le debolezze della infrastruttura.

4.3 Principio della globalità

"Una catena è forte quanto il suo più debole anello"

Un'infrastruttura informatica complessa è composta da numerosi elementi strettamente interconnessi.
La sicurezza dell'intera infrastruttura è il risultato della sicurezza dei singoli elementi e, soprattutto, della sinergia che i singoli elementi, una volta raggruppati, riescono a formare. Non ha senso rafforzare massicciamente la sicurezza di un solo elemento lasciandone vulnerabile un altro: in tal caso, chi compie la frode informatica sfrutterà l'insicurezza di quest'ultimo per violare la sicurezza dell'intero sistema.
Chi è intenzionato a violare la sicurezza del sistema cercherà di passare per la strada più breve, cioè per quella con il più conveniente rapporto costi / benefici. Spesso la via più facile per accedere illegalmente alle informazioni non è affatto tecnica.
Talvolta è preferibile, per l'attaccante acquisire le informazioni che desidera corrompendo un addetto interno piuttosto che tentando un attacco tecnico ad alta sofisticazione come la crittoanalisi di un algoritmo crittografico con cui sono protetti i dati.

4.4 Principio dell'unico punto di contatto

"È più facile controllare un unico punto di passaggio"

È buona norma concentrare le funzioni di sicurezza applicative e di rete su di un numero esiguo di sistemi, in maniera che la sicurezza dell'intera infrastruttura dipenda da pochi punti altamente controllabili.

4.5 Principio della modularità

"È più facile controllare la sicurezza di piccoli oggetti"

Oggetti piccoli sono più facilmente gestibili e controllabili. Nel caso che un oggetto fallisca, la sicurezza dell'intera infrastruttura può essere preservata. Un oggetto piccolo, inoltre, ha una complessità inferiore rispetto ad un oggetto grande e integrato ed è quindi più difficile che al suo interno siano contenute debolezze applicative (*bugs*). Questo principio permette anche di individuare con maggiore facilità le parti più critiche del sistema, dando la possibilità di interventi il più possibile mirati nell'evenienza di aggiunte, potenziamenti o aggiornamenti di ciascuna delle componenti.

4.6 Principio della ben definita politica di sicurezza

"Nel dubbio, meglio negare che permettere"

Nella progettazione di un sistema di sicurezza sono possibili due approcci:

1. Quello che non è espressamente permesso è proibito
2. Quello che non è espressamente proibito è permesso

In linea generale, il primo approccio è sempre preferibile dal punto di vista della sicurezza.

4.7 Principio della semplicità

"KISS: Keep It Simple Stupid"

La semplicità va d'accordo con la sicurezza. Ma complessità va d'accordo con la mancanza di visibilità da cui, immancabilmente, scaturisce l'insicurezza. Le componenti di un sistema di sicurezza devono essere il più semplici possibili, affinchè il sistema risulti facile da usare e da gestire. È un errore storico quello di pensare che un sistema grande e complesso debba essere sicuro. Un sistema grande e complesso è tipicamente difficile da analizzare, fino a diventare *oscuro.*

Quello che per noi è difficile da capire può apparire cristallino agli occhi di chi vuole compiere una frode informatica. La semplicità, quindi, gioca dalla nostra parte: più un oggetto è semplice, più una procedura è comprensibile e maggiori sono le probabilità che sia sicura. È noto dall'ingegneria del software che i programmi complessi hanno più *bugs* e tra questi è probabile che ce ne siano alcuni relativi alla sicurezza[1].

[1] Alcuni studi dimostrano che, in fase di sviluppo di codice, ogni circa 200 righe viene introdotto un *bug*.

4.8 Principio di Kerchhoff

"Chi compie la frode conosce sempre tutti i dettagli implementativi"

Se la robustezza di un sistema di sicurezza è basata sul fatto che non siano pubblicamente noti gli *internals*, gli algoritmi o le specifiche tecnologie usate, allora il sistema in questione è assai insicuro. È un approccio errato credere che, al fine di aumentare la sicurezza, sia meglio mantenere la propria tecnologia di difesa segreta piuttosto che lasciare che tale tecnologia venga visionata da un grande numero di esperti. Assumere che sia un compito difficile effettuare il *reverse engineering* di un'applicazione è un grave errore, un errore che purtroppo viene commesso da molti. I migliori oggetti di sicurezza sono quelli che impiegano algoritmi e protocolli pubblici che sono stati attaccati, analizzati e corretti per anni dai migliori esperti di sicurezza. È storicamente noto come moltissimi prodotti definiti *proprietary* sono risultati del tutto insicuri ed inadeguati una volta che i loro *internals* sono stati scoperti e resi pubblicamente noti.