UNIVERSITÀ DEGLI STUDI DI PERUGIA

Facoltà di Scienze Matematiche Fisiche Naturali
Corso di Laurea Magistrale in Informatica

Sicurezza Informatica
Port Knocking

Studente
Andrea Di Saverio
Docente
Stefano Bistarelli

Anno Accademico 2010-2011

Indice

Introduzione

Come ben noto il problema della sicurezza informatica non ha una soluzione unica e assoluta: le moderna teoria affronta il problema secondo un approccio multi-livello, comunemente definito "a cipolla". Combinando varie tecniche, che usate singolarmente potrebbero non rivelarsi sufficienti a garantire un adeguato livello di sicurezza, si ottiene un livello finale accettabile.

In prima approssimazione la sicurezza di una macchina che offre servizi su una rete pubblica dipende anche da quanti e quali sono i servizi stessi. L'uso di un firewall, la prontezza nell'installazione di patch e il tenersi sempre aggiornati sui siti di sicurezza informatica è il minimo che si possa fare, ma come l'esperienza insegna a volte non basta. Se per la maggior parte dei servizi non si può fare di meglio, per quelli che possono essere avviati solo quando necessario, e non sono quindi destinati al pubblico di massa, è invece possibile aggiungere un'ulteriore protezione tramite una tecnica chiamata **port knocking**.

Capitolo 1

Port Knocking

1.1 Generalità

Sui server connessi a reti pubbliche è spesso necessario mantenere servizi aperti per la gestione da remoto del server stesso o per altre finalità: un esempio su tutti è rappresentato dal protocollo SSH (Secure SHell) che permette di creare una connessione remota su un canale cifrato. Lasciare aperto un servizio simile può rappresentare un invito, per un potenziale attaccante, a penetrare nel sistema.

Servizi come questi, con una base di utenti limitata, non hanno bisogno di essere costantemente esposti al pubblico. A differenza di altri protocolli come SMTP o HTTP, che in genere non necessitano di autenticazione, i servizi di cui parliamo richiedono generalmente delle credenziali per l'accesso. Perché allora si rende necessario un ulteriore livello di sicurezza? Perché lasciando inaccessibile la porta del nostro servizio la proteggiamo anche dall'applicazione di *exploits*: ciò può essere sufficiente a "coprire le spalle" del nostro server in quel lasso di tempo che spesso intercorre tra la scoperta di un nuovo *bug* e il rilascio di una *patch*.

Come fare? La soluzione possono essere infinite: potremmo predisporre un timer che apre il nostro servizio per un tempo limitato, ad esempio per 1 minuto, una volta

ogni 30 minuti. Questa soluzione funziona ma comunque non risolve il problema: non è efficace poichè limita la nostra "operabilità" e al contempo, anche se per poco tempo, lascia comunque il servizio scoperto. Un'altra soluzione potrebbe essere quella di avviare il servizio quando il server riceve un nostro sms: in questo caso la soluzione è più efficace, ma forse troppo macchinosa e dispendiosa.

Ciò di cui si necessita è la possibilità di farsi riconoscere in maniera semplice e meno invasiva possibile. Anche mettere in ascolto un altro servizio su un'altra porta non rappresenta una soluzione buona: in questo caso non faremmo altro che spostare il problema. Una buona soluzione è invece rappresentata dal **Port Knocking**.

1.2 In teoria

Il *Port Knocking* è una tecnica che permette di comunicare con una macchina remota, e quindi di autenticarci, tramite l'invio di pacchetti SYN verso predefinite porte (filtrate dal firewall) in un preciso ordine. La **sequenza di knock** rappresenta la chiave di autenticazione.

Come esempio consideriamo una macchina alle cui porte 1111, 2222, 3333 non è associato alcun servizio. Se su questa macchina il firewall è configurato per monitorare l'arrivo di pacchetti a queste porte allora essa sarà in grado di riconoscere la sequenza: non importa se il client ha ricevuto un "*connection refused*". In realtà il server ha ricevuto il messaggio, ci ha riconosciuto, e ci aprirà il servizio.

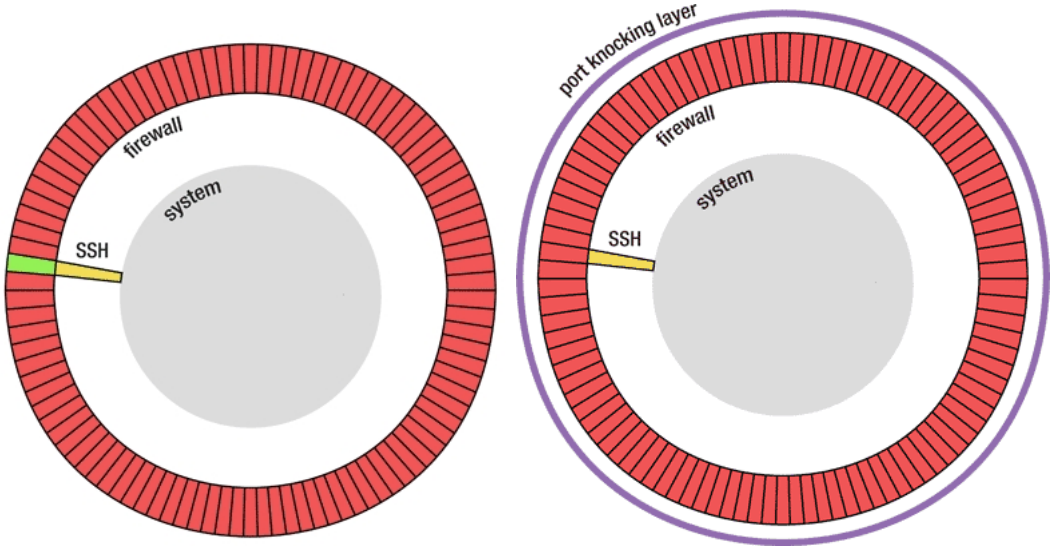
Tornando all'esempio del servizio SSH è facile intuire come questa tecnica permetta di superare alcuni "limiti" delle sue più comuni implementazioni: in genere nel gestire la sicurezza di questo servizio viene stilata una lista di indirizzi IP *trusted* a cui è concessa la connessione. Ciò ha però due limiti: innanzitutto il fatto che gli indirizzi IP non sempre sono statici, e poi il fatto che dietro un indirizzo IP fidato non è detto stia operando un utente *trusted*. Il *Port Knocking* invece permette a

uno specifico utente di collegarsi da qualsiasi indirizzo IP, piuttosto che permettere a qualsiasi utente di collegarsi da uno specifico indirizzo IP.

1.3 Security through obscurity

Quella descritta potrebbe quindi sembrare una forma di sicurezza tramite segretezza, in cui l'integrità del sistema è garantita dal fatto che l'attaccante manca di conoscenza sul sistema vittima. In realtà non è così, perché la sicurezza si basa sulla conoscenza di una sequenza di knock, una sorta di *shared-key*, e non sul sistema difensivo implementato. La forma di segretezza che garantisce questa tecnica riguarda invece il servizio attivo sulla macchina.

Il *port knocking* aggiunge un ulteriore strato al modello "a cipolla" senza rimpiazzare quelli precedenti (vedi Figura 1.1): anche se questo strato venisse meno la macchina (e nel nostro caso il servizio SSH) è comunque protetta dai propri meccanismi già implementati.

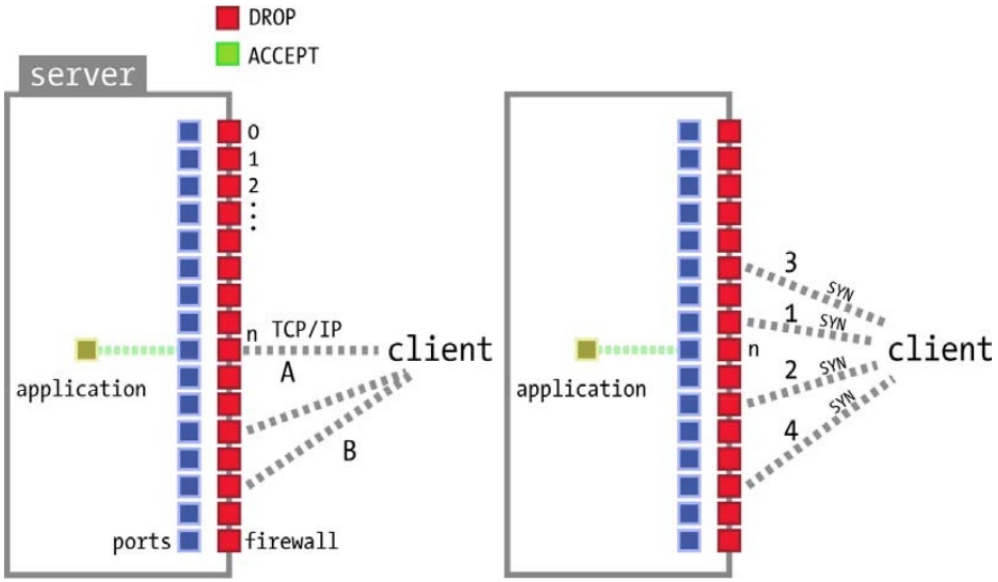


(a) (b)

Figura 1.1: Modello a cipolla: sistema classico (a) e con port knocking (b)

1.4 Funzionamento

Come già accennato il port knocking è una forma di autenticazione passiva nel cui processo non vi è comunicazione da parte del server. Inzialmente (Figura 1.4 (a)) il client tenta la connessione al servizio in ascolto sulla porta n: il firewall blocca la connessione e non restituisce (DROP) alcuna risposta al client, in modo da non fornire alcuna ulteriore informazione. Tutti i tentativi di connessione da parte del client non vanno a buon fine poiché rifiutati dal firewall.

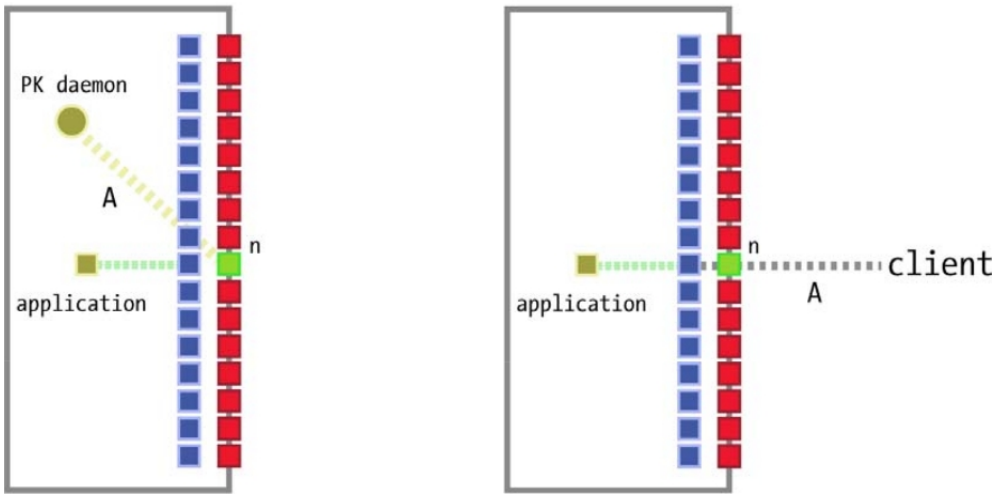


(a) (b)

Successivamente il client (Figura 1.4 (b)) prova a connettersi a un ben definito set di porte in un preciso ordine, inviando ad esse dei pacchetti SYN.

Un processo sul server (un port-knocking daemon) intercetta (Figura 1.4 (c)) i precedenti tentativi di connessione e li interpreta come una corretta sequenza di knock. A questo punto esegue i compiti associati a quella specifica sequenza e modifica la regola del firewall per aprire la porta n al client che ne ha fatto richiesta.

Finalmente (Figura 1.4 (d)) il client può connettersi al server tramite la porta n ed autenticarsi al servizio tramite i regolari meccanismi.



(c) (d)

1.5 In pratica

L'idea può essere implementata in molti modi: tramite pacchetti UDP, TCP, con aggiunta di dati e chiavi al pacchetto o in tanti altri modi. In ogni caso il server dovrebbe rivestire un ruolo passivo, e non fornire alcun feedback al client, cosa che spingerebbe un utente ben motivato ad approfondire tramite portscan o bruteforcing.

Il modo più semplice per implementare un sistema del genere è quello di sfruttare il proprio firewall affinchè questo logghi ogni tentativo di connessione. Tramite un parser verrà poi verificato se nel file di log è presente la sequenza sbloccante, e in tal caso verrà aperta la porta.

Anche per richiudere il servizio appena aperto sono possibili più alternative: la più semplice di tutte è la chiusura automatica dopo un certo tempo di inattività su quella porta.

1.6 Vulnerabilità e soluzioni

Nella possibile implementazione appena descritta sono numerose le vulnerabilità di questa tecnica. Vediamo le più lampanti:

1.6.1 Replay Attack

Quello che facciamo nel momento in cui "bussiamo" al nostro server è inviargli in chiaro l'equivalente di una password. Un attaccante che sniffa il nostro traffico non ha bisogno di conoscere il meccanismo di difesa utilizzato: ogni volta che viene effettuata una connessione SSH rileva che precedentemente vengono inviati sempre i soliti pacchetti, alle solite porte, e con esito fallimentare. A questo punto è sufficiente replicare lo stesso traffico per ottenere lo stesso risultato.

Una possibile soluzione è l'utilizzo di sequenze di knock "usa e getta": una volta utilizzata una sequenza questa viene automaticamente modificata e comunicata al client, come una sorta di *one time password*. Una soluzione migliore è l'utilizzo della crittografia: il client, nel segmento *data* del pacchetto, invia un *nonce* crittografato con una shared-key nota a lui e al server. Il server non accetterà ulteriori sequenze con lo stesso nonce.

1.6.2 Man In The Middle Attack

In questo tipo di attacco l'attaccante si interpone nella comunicazione tra client e server, con la possibilità di modificare il messaggio che dal client va verso il server. Sarebbe dunque suffciente all'attaccante, una volta intercettato il traffico del client, inoltrarlo con l'indirizzo IP sorgente corrotto, fornendo così a se stesso l'apertura del servizio.

Anche in questo caso si rende necessario l'uso della crittografia per scongiurare il problema: nel segmento *data* del pacchetto potremmo aggiungere, crittografa-

to, l'indirizzo ip del client. Il server verificherà che questo corrisponde all'indirizzo sorgente: se così non fosse scarta il pacchetto e le richieste.

1.6.3 DoS

Un attacco di tipo *Denial of Service* potrebbe comprottere il funzionamento del demone causandone addirittura l'interruzione. In questo caso correremmo il rischio di rimanere tagliati fuori dalla macchina server: questo rappresenta dunque un **single point of failure**.

Per evitare simili inconvenienti è necessario implementare un servizio di *monitoring* sulla macchina, che controlla costantemente l'esecuzione del demone e si occupa del suo eventuale riavvio. A volte potrebbe rivelarsi consigliabile lasciare in *allow* un indirizzo IP sicuro da cui effettuare la connessione, in modo da potersi comunque autenticare anche se qualcosa del meccanismo di port knocking dovesse andare storto.

1.6.4 Brute forcing

Dato che le porte disponibili sono $2^{16} = 65536$ le possibili combinazioni sono 65536^n, dove n è il numero di knock nella sequenza: per n adeguato il risultato è un numero sufficientemente grande da scoraggiare simili tentativi. A questa considerazione si può associare un ulteriore comportamento del demone: dopo un numero predefinito di tentativi errati, l'indirizzo IP del richiedente viene definitivamente escluso.

Conclusioni

Quella del Port Knocking è una tecnica relativamente nuova che permette, in maniera semplice e sfruttando strumenti generalmente già installati su una macchina, di aggiungere un ulteriore livello di sicurezza al sistema. Questo non deve essere considerato un metodo alternativo ai meccanismi di difesa già esistenti, ma semmai un di più che sicuramente non guasta.

Anche se non era scopo di questo lavoro mostrare le soluzioni e le implementazioni esistenti, queste sono molte e in costante sviluppo: ogni soluzione presenta proprie caratteristiche e personali approcci al problema, dall'uso della crittografia a quello di sequenze mutevoli, ma comunque in tutti i casi sono semplici da installare e configurare.

Bibliografia

[1] http://www.portknocking.org/docs/portknocking_an_introduction.pdf

[2] http://www.linuxjournal.com/article/6811

[3] http://en.wikipedia.org/wiki/Port_knocking