Un’analisi dei problemi del WEP

In numeri precedenti di questa rivista sono già apparsi vari articoli che si sono occupati dei problemi di sicurezza del protocollo 802.11, detto anche WiFi, ed in particolare del protocollo di crittografia adottato dall’802.11, il WEP (Wired Equivalent Privacy). Ad esempio è di particolare interesse l’articolo di Luigi D’Amato nel numero di Gennaio 2003. Inoltre in una serie di articoli nella rubrica Hands−On (Dicembre 2002 − Febbraio 2003) abbiamo fatto una introduzione tecnica al protocollo 802.11 che è preliminare a questo articolo. In questo articolo vogliamo infatti approfondire la discussione tecnica del protocollo WEP e dei suoi problemi, anche perché ci sembra che questa sia una storia dalla quale si possono trarre alcuni insegnamenti per il futuro.

1. Sicurezza e Protocolli Wireless

Vogliamo qui solo ricordare come i protocolli Wireless abbiano ulteriori problemi di sicurezza rispetto alle loro controparti su cavo. Ad esempio vi sono i seguenti tipi di utilizzo fraudolento dei protocolli Wireless:

1. Se una rete ha una componente Wireless, è possibile che un attaccante dall’esterno dell’edificio possa ottenere copia almeno di tutto il traffico wireless, oppure introdursi nella rete ed utilizzarla per connettersi ad internet o come punto di partenza per effettuare attacchi ad altri;
2. Un malintenzionato potrebbe inserire un piccolo AccessPoint (AP) wireless in una rete cablata ed ottenere copia del traffico che vi passa standosene tranquillamente nell’edificio di fronte.

Ovviamente nel secondo caso il malintenzionato deve prima introdursi nell’edificio per installare l’AP, ma purtroppo questa è una cosa di solito non molto difficile.

In questo articolo ci occupiamo invece di cosa l’802.11 dovrebbe fare per prevenire il primo problema. A prima vista a chiunque sembrerebbe che sia necessario:

a) che vi sia un protocollo sicuro di autenticazione ed autorizzazione, per evitare che una stazione possa connettersi alla rete senza avere le necessarie credenziali;
b) che tutto il traffico tra Stazioni ed AccessPoint sia cifrato, per evitare che qualcuno in ascolto possa fare una copia di ciò che transita.

Il WEP in pratica non affronta il primo punto, mentre l’implementazione del secondo è risultata sbagliata. La prima domanda ovvia è: perché è stato rilasciato questo protocollo se si sapeva sin dall’inizio che il primo punto non era stato trattato a sufficienza?

2. L’autenticazione WEP

L’idea dell’autenticazione WEP è molto semplice. Su ogni Stazione e AccessPoint si inserisce una

(o più per un AccessPoint) chiave WEP (a 40 o 104 bit). Nessun protocollo è previsto per l'aggiornamento o manutenzione delle chiavi, che pertanto di solito sono inserite manualmente nelle schede WiFi e negli AccessPoint. In pratica non è raro che anche in grandi reti WiFi tutte le stazioni ed AccessPoint abbiano la stessa chiave WEP che non viene mai cambiata. Avere la corretta chiave WEP è in pratica l'unica misura di autenticazione WEP di una Stazione verso un AccessPoint.[1] Inoltre un vero protocollo di autorizzazione non è specificato. Questi problemi dovrebbero essere risolti con l'introduzione dell'802.1x e dell'EAP (Exstensible Authentication Protocol).

Va sottolineato che non è semplice introdurre nuovi protocolli oggi. Infatti già milioni di device sono in uso ed è necessario che i nuovi protocolli e le nuove funzionalità siano compatibili con l'hardware esistente, per ovvie ragioni economiche. Ma l'hardware esistente ha risorse molto limitate, solo il traffico può utilizzare sino al 90% della CPU di un AccessPoint ed i protocolli di crittografia e sicurezza devono accontentarsi di quello che resta.

3. L'implementazione di RC4

Come detto, WEP cifra tutto il traffico tra una stazione e l'AccessPoint. L'algoritmo utilizzato è l'RC4 [5], uno *Stream Cipher* ideato da Ron Rivest dell'RSA Security. L'algoritmo di per se è sicuro, veloce e abbastanza leggero da implementare. Uno Stream Cipher, data una chiave iniziale segreta di lunghezza fissa, può generare un flusso *infinito* e pseudo−random di bit che formano la chiave di cifratura. In pratica il procedimento di cifratura è il seguente.

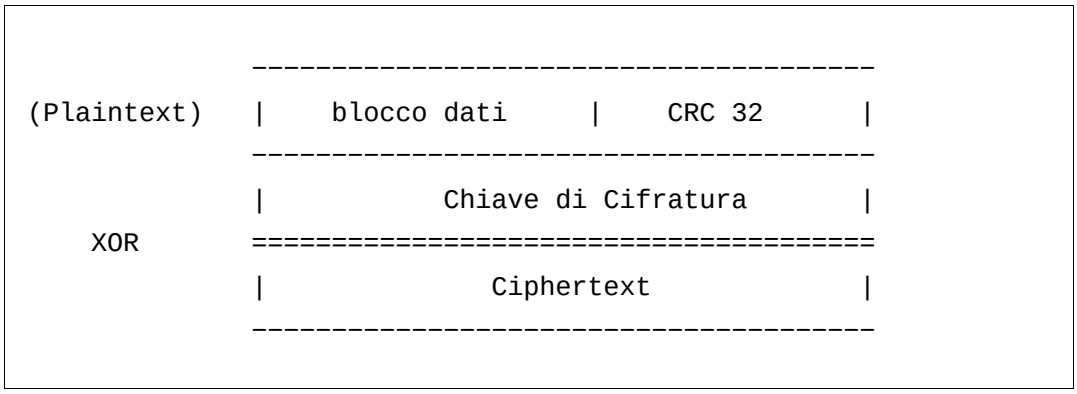
Alla chiave (segreta) WEP di 40 o 104 bit viene aggiunto un vettore di inizializzazione (IV) di 24 bit in modo da formare una chiave intermedia di 64 o 128 bit. L'IV è un numero che viene generato dall'AccessPoint o dalla Stazione, spesso è un registro che viene incrementato di 1 ogni volta che viene utilizzato. In questo modo si possono generare 2^24 (quasi 17 milioni) chiavi intermedie. La chiave intermedia è l'input dell'algoritmo RC4 che genera una Chiave di Cifratura (o Stream) di lunghezza fissa:

```
Chiave_Intermedia = Chiave_WEP ∪ IV
Chiave_Cifratura = RC4(Chiave_Intermedia)
```

Contemporaneamente i dati da cifrare vengono divisi in blocchi di lunghezza fissa, e di ogni blocco viene calcolata una checksum CRC a 32 bit che viene aggiunta in coda al blocco. Il blocco di dati più la checksum forma il Plaintext ed ha la stessa lunghezza della Chiave di Cifratura. Nell'ultimo passaggio si esegue un XOR del Plaintext con la chiave di cifratura per ottenere il

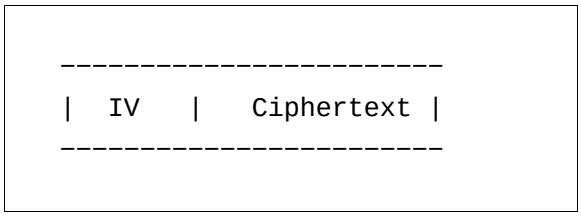
1 Ovviamente l'autenticazione di base, SSID e MAC−address ad esempio, deve essere valida.

Ciphertext, o messaggio cifrato:



Utilizzeremo anche la seguente notazione matematica: $P = D \cup CRC$, ove P è il Plaintext, D il blocco dati, CRC la checksum CRC−32 e $\cup$ l'unione dei due pacchetti; $C = P \otimes K$, ove K è la Chiave di Cifratura, C il Ciphertext e $\otimes$ l'XOR.

L'XOR è una operazione che dati 2 bit, ad esempio uno del Plaintext ed il corrispondente della Chiave di Cifratura, se sono uguali produce il bit 0, se sono diversi il bit 1. Si noti che l'XOR di un numero con se stesso da come risultato tutti 0 e l'XOR di zero con un qualsiasi altro numero da sempre di nuovo il secondo numero. Infine si invia sul network un pacchetto composto dall'IV più il Ciphertext:



Il procedimento di decrittazione è l'esatto opposto: mettendo insieme la chiave (segreta) WEP presente nel device e l'IV ottenuto dal pacchetto stesso, utilizzando RC4 si genera la stessa Chiave di Cifratura. Facendo l'XOR del Ciphertext con la Chiave di Cifratura, che serve quindi anche per de−cifrare, si ottiene il Plaintext ($K \otimes C = K \otimes (P \otimes K) = P$). Si controlla se la checksum CRC−32 è corretta e nel caso si accetta il pacchetto.

Bisogna notare che l'IV è inviato in chiaro nel pacchetto, e non potrebbe essere altrimenti poiché il ricevente in caso contrario non potrebbe calcolarsi la Chiave di (de−) Cifratura. D'altronde la sicurezza di RC4 garantisce che un attaccante in possesso solo di un IV e del corrispondente Ciphertext, non è in grado di risalire alla Chiave di Cifratura. L'IV garantisce che ogni pacchetto è crittato con una chiave intermedia diversa, e questo è uno dei punti fondamentali dell'algoritmo.

D'altra parte, se lo stesso IV fosse usato più volte con la stessa chiave WEP, l'algoritmo non sarebbe più sicuro. Vediamo come. Supponiamo di crittare due blocchi di dati con lo stesso IV e la stessa chiave WEP, e quindi con la stessa Chiave di Cifratura. Prendiamo ora i due pacchetti cifrati

e facciamone l'XOR. Quello che otteniamo è interessante:[2]

$$C1 \otimes C2 = (P1 \otimes K) \otimes (P2 \otimes K) = P1 \otimes P2$$

ove C1, C2 sono i Ciphertext e P1, P2 i Plaintext. Il risultato è che abbiamo eliminato la Chiave di Cifratura ed ottenuto l'XOR dei Plaintext. Ora se un attaccante fosse in grado di conoscere uno dei due Plaintext, sarebbe in grado di ottenere l'altro con un semplice XOR

$$P1 \otimes (C1 \otimes C2) = P1 \otimes (P1 \otimes P2) = P2$$

Non solo, conoscendo un Plaintext ed il corrispondente Ciphertext, si può ottenere di nuovo con un XOR la Chiave di Cifratura

$$P1 \otimes C1 = P1 \otimes (P1 \otimes K) = K$$

Conoscendo alcune Chiavi di Cifratura K ed i relativi IV, sfruttando una debolezza dell'algoritmo RC4 scoperta da Fluhrer, Mantin e Shamir [1], con un po' di sforzo ulteriore e l'utilizzo di tecniche di criptoanalisi differenziale, si può risalire alla chiave WEP [2,3] e così rompere completamente la sicurezza.

Purtroppo tutto questo capita in WEP. Infatti come abbiamo detto, l'IV è un numero a 24 bit, e quindi permette di creare solo circa 17 milioni di chiavi intermedie. Ipotizzando di inviare pacchetti di 1500Bytes a 11Mbps, ognuno con un IV diverso, bastano poco più di 5 ore per esaurire gli IV. In pratica però ci vuole molto meno, poiché ad esempio in caso di errori di trasmissione (collisioni ecc.) il protocollo WEP richiede la reinizializzazione degli IV. Se poi gli IV fossero dati in modo casuale, ci vorrebbe ancora meno per avere una ripetizione. Inoltre non è necessario conoscere completamente un Plaintext o fare un *known plaintext attack*, infatti sfruttando gli header dei pacchetti, che sono spesso facilmente indovinabili, ed i pacchetti tipo ARP, IP, SNAP ecc. che spesso si ripetono nella rete, è possibile ottenere sufficienti informazioni per ricostruire la Chiave di Cifratura. Il tutto è così semplice, che esistono programmi come **Airsnort** e **Wepcrack** che rendono del tutto automatico l'ottenimento della chiave (non più segreta a questo punto) WEP. Se la chiave WEP è di 40 bit, poche ore di ascolto su di una rete WiFi (con sufficiente traffico) permettono di ottenere la chiave stessa. Se la chiave WEP è invece di 104 bit, molte più coppie (Chiave di Cifratura, IV) e molte più ore sono necessarie per risalire

2 Ricordiamoci che l'XOR di un numero con se stesso fa zero, e zero in questo caso agisce da identità.

alla chiave WEP.

A parziale scusante dell'IEEE e della WiFi Alliance, bisogna dire che se in [1] non fosse stata scoperta questa debolezza di RC4, sarebbe molto più difficile, se non quasi impossible, fare l'ultimo passo per ottenere la chiave WEP. D'altronde, con una corretta gestione dell'IV e delle chiavi, RC4 è ancora un protocollo sicuro. Se sin dall'inizio il WEP avesse previsto la rigenerazione frequente della chiave segreta iniziale di crittazione, questo problema non sarebbe sorto.

4. I problemi di CRC−32

Un altro problema, veniale a paragone del precedente, che è stato trovato in WEP è nell'utilizzo di CRC−32. CRC, Cyclical Redundacy Check, è un algoritmo di checksum che permette di calcolare un numero di lunghezza fissa, 32 bit in questo caso, caratteristico del pacchetto inviato. La checksum viene calcolata prima dell'invio e della crittazione dei dati, ed inserita nel pacchetto; all'arrivo viene ricalcolata e confrontata con quella presente nel pacchetto. Se vi è una differenza vuol dire che il pacchetto è stato modificato in qualche modo, e quindi viene scartato. Il problema principale con CRC−32 [3] è che è una checksum lineare, ed è stato trovato il modo di modificare un bit nella parte dati del pacchetto ed un bit nella checksum in modo che i nuovi dati corrispondano alla nuova checksum, il tutto all'interno del pacchetto crittato con WEP! Anche se ovviamente di entità minore rispetto al problema precedente, il fatto che un attaccante sia in grado di modificare un pacchetto crittato e che questo venga accettato dall'algoritmo, è una grave lacuna del protocollo.

5. Conclusioni

Molto è stato scritto su come proteggersi dai mali del WEP. Purtroppo tutte le statistiche recenti indicano che la grande maggioranza delle reti WiFi non usa neanche il WEP, e quindi sono aperte completamente ad ogni tipo di attacco. Anche se il WEP è stato rotto, utilizzarlo, e con chiavi a 104 bit se possibile, è meglio di nulla. Almeno con il WEP attivo l'attaccante non avrà accesso istantaneo alla rete e molto probabilmente utilizzerà un'altra rete non protetta. Nei prossimi mesi i nuovi protocolli WPA dovrebbero risolvere i problemi del WEP, mentre per il momento è conveniente cifrare tutto il traffico a monte utilizzando ad esempio delle VPN IPSEC.

L'unica morale che vogliamo trarre dalla storia del WEP, è che quando si tratta di sicurezza, le scorciatoie non sono mai convenienti e che sono sempre necessari molteplici livelli di protezione per evitare che la caduta di un punto faccia crollare tutto il sistema. Il WiFi ha avuto la sfortuna

che la debolezza di RC4 [1] è stata scoperta troppo tardi, e la decisione di non includere la gestione delle chiavi nel protocollo si è rivelato un errore.

(L'autore ringrazia Fabrizio Croce, Country Manager Italy WatchGuard Technologies, fabrizio.croce@watchguard.com, per il contributo dato alla realizzazione di questo articolo.)

Andrea Pasquinucci
Consulente di Sicurezza Informatica
pasquinucci@ucci.it

Riferimenti Bibliografici:

[1]S. Fluher, I. Mantin, A. Shamir, *Weaknesses in the Key Scheduling Algorithm of RC4*, http://online.securityfocus.com/data/library/rc4_ksaproc.pdf

[2]A. Stubblefield, J. Ioannidis, A.D. Rubin, *Using the Fluher, Mantin, Shamir Attack to break WEP*, http://www.cs.rice.edu/~astubble/wep/wep_attack.pdf

[3]N. Borisov, I. Goldberg, D. Wagner, *Intercepting Mobile Communications: the Insecurity of 802.11*, http://www.isaac.cs.berkeley.edu/isaac/wep−draft.pdf

[4]M. Sutton, *Hacking the Invisible Network,* iDefense, http://www.idefense.com/

[5]Per l'RC4 si veda ad esempio B. Schneier, *Applied Cryptography*, John Wiley 1996; per una introduzione matematica agli Stream Cipher si veda anche A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press 1996.