Italian Black Hats Speech

(INFOSECURITY ITALIA 2002)

Italian Black Hats Association

http://www.blackhats.it

Sovversioni del protocollo TCP/IP: attacchi e contromisure

Milano, 24/1/2002, Sala Cadamosto

Copyright

INDEX

- TCP, introduzione al protocollo
- Numeri di Sequenza
- Sniffing
- RAW Sockets
- Analisi e Predizione dell'ISN
- TCP Spoofing
- TCP Hijacking
- Man in the Middle
- TCP Mangling
- Riferimenti e Documentazione

SPEAKER

Matteo Falsetti aka fusys

Non verranno presi in esame tutti gli algoritmi alla base di TCP, nè tutti i possibili iter decisionali che lo stack possa seguire per far fronte a situazioni differenti.

L'accento viene messo sulle caratteristiche basilari e sul loro utilizzo da parte di un attaccante.

La seguente presentazione non vuole assolutamente sostituire la lettura di RFC o di pubblicazioni specificatamente mirate all'introduzione di TCP.

Inoltre si considererà l'auditorio in possesso di nozioni di base sul funzionamento del protocollo.

TCP, Transmission Control Protocol

L'RFC con status STANDARD è l'RFC793, datato Settembre 1981, basato su nove precedenti edizioni del draft.

Buon Compleanno Nonno ;-) ! Il protocollo TCP ha da poco compiuto 20 anni, ufficialmente. Difficile decidere se parlare di maturita' o di vecchiaia.

Il protocollo è stato man mano esteso, non rivisto.

Decine di documenti tecnici hanno presentato in venti anni svariati problemi inerenti il cuore stesso del protocollo

TCP, Transmission Control Protocol

Dall'introduzione dell'RFC793:

"Il Transmission Control Protocol (**TCP**) è usato come sistema altamente affidabile di protocollo punto-punto tra sistemi in reti di computer a commutazione di pacchetto, ed in sistemi interconnessi di reti suddette"

TCP, Transmission Control Protocol

"**TCP** è un affidabile protocollo orientato alla connessione, termino-terminale, architettato per inserirsi in una gerarchia stratificata di protocolli che supportano applicazioni multi-rete. **TCP** provvede alla comunicazione tra processi tra coppie di applicazioni in sistemi di computer collegati a reti di comunicazione informatiche distinte, ma interconnesse"

TCP, Transmission Control Protocol

"**TCP** si inserisce in una architettura di protocolli stratificati, appena sopra un semplice Protocollo Internet (**IP**).“

"Il Protocollo Internet porta anche informazioni sulla precedenza, la classificazione di sicurezza e la compartimentazione dei segmenti **TCP**“

Dall'RFC791, "Internet Protocol" :

"Sicurezza. Questa opzione fornisce un modo, per i sistemi, di inviare parametri di sicurezza, compartimentazione, restrizioni d'uso e TCC o Transmission Control Codes."

Esempio di opzione IP relativa alla sicurezza [IPOPT_SECURITY]:

Type=130 Length=11
Security (S field): 16 bits
Specifica uno tra 16 livelli di sicurezza (8 dei quali sono riservati per uso futuro)

00000000 00000000 - Unclassified
11110001 00110101 - Confidential
01111000 10011010 - EFTO
10111100 01001101 - MMMM
01011110 00100110 - PROG
10101111 00010011 - Restricted
11010111 10001000 - Secret
01101011 11000101 - Top Secret

TCP, Transmission Control Protocol

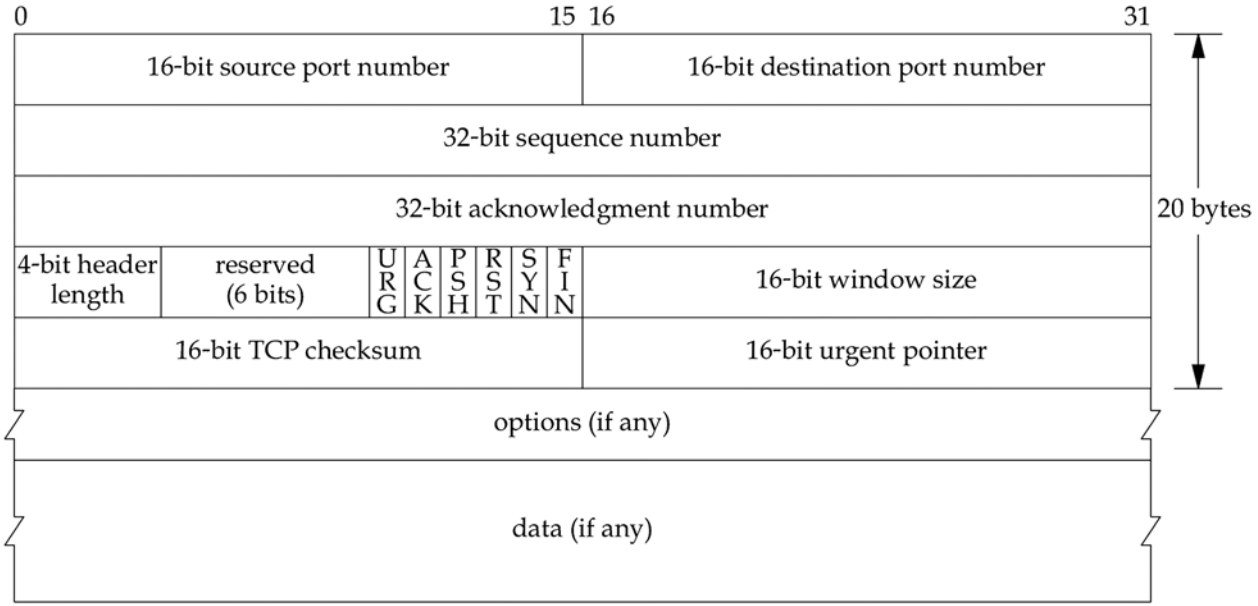
Un attaccante può costruire un header IP funzionale e conforme agli standard, a seconda delle proprie esigenze. Anche questa opzione può essere forgiata a piacimento.

Spesso comunque le opzioni IP non sono passate dai router, o non sono gestite dagli stack TCP/IP di default.

Quindi, **TCP** è da considerarsi ***insicuro*** di default, almeno dal nostro punto di vista.

TCP, Transmission Control Protocol

TCP Header



TCP, Transmission Control Protocol

Parametri Importanti per l'Attaccante

Source Port	la porta sorgente, campo a 16bit
Destination Port	la porta destinazione, campo a 16bit
Sequence Number	Il numero di sequenza del primo ottetto di dati (tranne quando sia presente la flag SYN). Se SYN è presente allora identifica l'ISN, o Initial Sequence Number ed il primo ottetto di dati è ISN+1
Acknowledgment Number	Se il bit di controllo ACK è attivo questo campo contiene il valore del prossimo numero di sequenza che aspetta di ricevere chi ha inviato il segmento. Qualora la connessione sia stabilita, è sempre inviato

TCP, Transmission Control Protocol

Le connessioni che utilizzano TCP sono identificate univocamente mediante la doppia coppia

[saddr, sport] <> [daddr, dport]

saddr indirizzo IP sorgente [contenuto nell'header IP]

sport porta sorgente TCP [Source Port]

daddr indirizzo IP destinazione [contenuto nell'header IP]

dport porta destinazione TCP [Destination Port]

Apertura di una Connessione TCP, o **3-Way Handshake**

TCP Client				TCP Server
1. CLOSED				LISTEN
2. SYN-SENT	-->	<SEQ=100><FLAG=SYN>	-->	SYN-RECEIVED
3. ESTABLISHED	<--	<SEQ=300><ACKN=101><FLAG=SYN,ACK>	<--	SYN-RECEIVED
4. ESTABLISHED	-->	<SEQ=101><ACKN=301><FLAG=ACK>	-->	ESTABLISHED

SEQ	Sequence Number
ACKN	Acknowledgment Number
SYN, **ACK**	Bit di Controllo SYN e ACK

Chiusura della Connessione TCP

Dal punto di vista dell'attaccante esistono due metodi per chiudere una connessione, quando invece l'RFC ne contempla solo uno, il primo:

Mediante bit di controllo **FIN**

Mediante bit di controllo **RST**

Chiusura mediante FIN

	TCP Client		TCP Server
1.	ESTABLISHED		ESTABLISHED
2.	(Close)		
	FIN-WAIT-1 -->	<SEQ=100><ACKN=300><FLAG=FIN,ACK>	--> CLOSE-WAIT
3.	FIN-WAIT-2 <--	<SEQ=300><ACKN=101><FLAG=ACK>	<-- CLOSE-WAIT
4.			(Close)
	TIME-WAIT <--	<SEQ=300><ACKN=101><FLAG=FIN,ACK>	<-- LAST-ACK
5.	TIME-WAIT -->	<SEQ=101><ACKN=301><FLAG=ACK>	--> CLOSED
6.	(2 MSL)		
	CLOSED		

SEQ Sequence Number

ACKN Acknowledgment Number

FIN, **ACK** Bit di Controllo FIN e ACK

Chiusura mediante RST

Questo metodo serve ad abortire una connessione in due specifiche situazioni.

Se la connessione non esiste, un RST dev'essere inviato in risposta ad ogni segmento in ingresso ad eccezione di un altro RST.

Se la connessione è in uno stato non sincronizzato (LISTEN, SYN-SENT, SYN-RECEIVED) ed il segmento in ingresso riconosce qualcosa di non ancora inviato (ovvero possiede un ACKN non accettabile) un RST dev'essere inviato.

Se il segmento in ingresso contiene un ACKN allora il RST otterra' il proprio SEQ da tale valore, altrimenti lo inizializzerà a 0. Il proprio ACKN sara‘ uguale alla somma del SEQ e della lunghezza del segmento in ingresso.

Chiusura mediante RST (2)

In tutti gli stati tranne SYN-SENT, tutti i segmenti RST sono convalidati mediante controllo del loro SEQ. Un RST è valido se il suo SEQ rientra nella finestra (TCP Window).

Nello stato SYN-SENT invece il segmento RST viene considerato accettabile se l'ACKN riconosce il SEQ inviato.

E' quindi possibile per un attaccante abortire una connessione a volontà, mantenendo il controllo sui valori di SEQ e ACKN.

Punti Deboli di TCP

TCP soffre sia di alcune debolezze intrinseche alla sua architettura interna, sia di problemi relativi alla sua implementazione. Questi ultimi sono ovviamente dipendenti dallo stack TCP/IP analizzato.

Il controllo della connessione si basa unicamente sui numeri di sequenza. Non esiste alcun'altra forma di autenticazione sulla sorgente dei segmenti

La generazione dei numeri di sequenza è passibile di predizione da parte dell'attaccante

TCP, Transmission Control Protocol

Numeri di Sequenza

Per stabilire una connessione lo stack TCP/IP utilizza l'ACKN del client. Se valida il primo SEQ (o ISN) del server allora la connessione viene portata a termine.

Per ritenere validi i dati inviati e ricevuti, dev'essere soddisfatta la relazione seguente:

SND.UNA < SEG.ACK =< SND.NXT

SEG.ACK e' il numero di sequenza atteso nel segmento in arrivo,
SND.UNA e' il piu' vecchio SEQ riconosciuto da un ACK e
SND.NXT e' il prossimo numero di sequenza da inviare.

Numeri di Sequenza (2)

Il metodo di generazione del primo Sequence Number del server, o Initial Sequence Number (ISN), non è necessariamente sicuro.

Il 25 Febbraio 1985 **Robert T.Morris**, dei laboratori AT&T Bell, pubblica un'analisi sommaria della possibilità di portare a termine connessioni con indirizzi IP contraffatti, non esistenti o di terze parti, dal titolo

"**A Weakness in the 4.2BSD Unix TCP/IP Software**"

TCP, Transmission Control Protocol

Numeri di Sequenza (3)

Nell'Aprile del 1989, **S.M.Bellovin** pubblica un documento dal titolo

"**Security Problems in the TCP/IP Protocol Suite**"

Il primo problema analizzato, "**TCP Sequence Number Prediction**", descrive nel dettaglio l'attacco teorizzato da Morris ben quattro anni prima.

Ma è solo nel Maggio 1996 che lo stesso Bellovin pubblica un RFC dall'ovvio titolo

"**Defending Against Sequence Number Attacks**“

dopo l'advisory del CERT in risposta al famoso caso **Mitnick**-**Shimomura**.

Numeri di Sequenza (4)

Secondo l'RFC793 il valore dell'ISN dovrebbe:

- essere subordinato ad un clock, la cui frequenza ne influenzerebbe l'incremento
- l'incremento dovrebbe essere di 125000 ogni mezzo secondo

Questo in realta' non avveniva. Dai sorgenti di BSD, da cui ogni primo stack TCP/IP ha preso forma, si evince che l'ISN:

- viene incrementato di 64000 ogni mezzo secondo
- per ogni nuova connessione portata a termine, viene incrementato di 64000, 128000 o relativo multiplo

Questo rese possibile il famoso attacco cui fanno riferimento **Morris**, **Bellovin**, **Shimomura** ed il famoso articolo presentato sul numero 48 della e-zine **Phrack**

TCP, Transmission Control Protocol

Numeri di Sequenza (5)

La soluzione proposta nell'**RFC1948** e' la seguente:

- associare ad ogni connessione un differente range di numeri di sequenza
- all'interno di ogni range utilizzare i normali incrementi come da RFC793
- usare una funzione di **hash crittografico** per computare l'ISN di ogni range, tale che

ISN = **M** + **F**(saddr, sport, daddr, dport)

M timer di 4 microsecondi
F hash mediante MD5, non computabile dall'esterno

Numeri di Sequenza (6)

I sistemi operativi **Linux** e **OpenBSD** sembrano essere gli unici a resistere ad analisi statistiche di una certa rilevanza, come quelle presentate nel corso del 2001 da **Guardent** e **Bindview**.

Dalla presentazione dell'RFC1948, la maggior parte dei sistemi operativi hanno modificato il sottosistema di generazione dell'ISN, eppure il problema non è risolvibile mediante il solo uso di generatori di numeri pseudocasuali, come mostrato nelle white paper di **Newsham** e **Zalewski**.

L'uso della crittografia per proteggere connessioni punto punto viene da loro auspicato e indicato come possibile soluzione valida.

Numeri di Sequenza (7)

"**The Problem with Random Increments**" presenta un'analisi matematica della generazione pseudocasuale, mentre "**Strange Attractors and TCP/IP Sequence Number Analysis**" mostra il comportamento dei valori SEQ in relazione alla teoria degli attrattori (peraltro citata da **Bellovin** nel suo RFC).

I risultati mostrano come la sola implementazione di **Linux** dell'RFC1948 sia da considerarsi valida, mentre quella della maggior parte dei cosiddetti UNIX commerciali sia affetta da problemi concettuali e/o implementativi.

Il metodo di hashing dell'RFC1948, se non coadiuvato da ulteriore casualità, presenta problemi non indifferenti data la natura dinamica della maggior parte degli IP in uso da parte dei client in InterNet.

Numeri di Sequenza (8)

L'approccio di attacco al protocollo TCP può avvalersi della possibilità di analizzare il traffico in corso, mediante compromissione di hop intermedi tra i due peer della connessione, o mediante alterazione delle tabelle di instradamento, rendendo così inutile e superflua ogni considerazione sui numeri di sequenza in questi scenari particolari.

Un esempio, datato 1995, è presentato nel documento

"**A Simple Active Attack Against TCP**"

Sovversione di TCP

Per portare a buon fine diversi attacchi sono necessarie per il cracker:

- tecniche di analisi passiva del traffico [**Sniffing**]
- tecniche di creazione di datagrammi IP [**RAW Sockets**]
- tecniche di **analisi** e **predizione** dell'**ISN**
- risorse computazionali e di banda, nei casi peggiori

Sniffing

Mediante uno **sniffer**, è possibile per l'attaccante visualizzare tutto il traffico di un segmento di rete.

Esistono strumenti per operare sniffing anche in ambienti di **Switched Lan**, mediante sovversione di protocolli di basso livello della suite TCP/IP, come ad esempio la corruzione della cache **ARP**.

Mediante tecniche combinate di sniffing e di implementazioni del concetto di **Man in the Middle**, è possibile analizzare il traffico di sessioni crittate, come sessioni **SSH** o **HTTPS**

Sniffing (2)

Uno dei più importanti strumenti per l'amministratore di rete, su sistemi Unix, è **tcpdump(1)**. Questo è un esempio di sniffer e delle sue potenzialità.

Mediante esso, è possibile per l'attaccante controllare lo stato dei numeri di sequenza, della tcp window, e del flusso dei dati in corso, in maniera interattiva.

```
12:00:50.668574 CLIENT.33044 > SERVER.25: S 1271811348:1271811348(0)
                                          win 32767 (DF) [tos 0x10]
12:00:50.668626 SERVER.25 > CLIENT.33044: S 1269949198:1269949198(0)
                                          ack 1271811349 win 32767 (DF)
12:00:50.668661 CLIENT.33044 > SERVER.25: . ack 1269949199 win 32767
                                          (DF) [tos 0x10]
```

Sniffing (3)

Risulta semplice per un attaccante costruire uno sniffer ad hoc, senza contare che esistono strumenti di buon livello tecnico facilmente reperibili in rete (**Sniffit**, **Ethereal**, **DSniff**, ...)

Un semplice sistema compromesso in LAN, con accesso di superutente, può ospitare uno sniffer. Il semplice uso di librerie come **libpcap**, o di banali chiamate come **socket(PF_PACKET, SOCK_RAW, htons(ETH_P_IP))** in Linux, permettono di accedere al livello datalink della stratificazione TCP/IP, e di ottenere copia del traffico di rete. La natura broadcast della tecnologia ethernet farà il resto.

Sniffing (4)

Nel caso l'attaccante abbia accesso al traffico di rete, ogni meccanismo di generazione dei Sequence Numbers diventa inutile. Sarà quindi possibile operare attacchi attivi contro lo stream TCP.

Il TCP Hijacking permette di sovvertire ogni connessione TCP, come dimostrato da diversi codici di attacco (ad es.: **Juggernaut**, **xTHOT**, **Hunt** ed **Ettercap**).

Esistono metodi che promettono di individuare sniffer all'opera in una LAN. Purtroppo sono fallaci nel momento in cui non possano far ricorso a peculiarità specifiche di alcuni kernel (v. **Antisniff** e patch Anti-Antisniff)

RAW Sockets

Gli stack TCP/IP permettono al superutente di creare datagrammi IP completi di ogni header di protocollo, in tal modo evitando il normale meccanismo di creazione e gestione di comunicazioni IP del kernel.

Questo è vero non solo per IPv4, ma anche per IPv6, mediante creazione, a livello più basso, anche del frame di datalink.

Le tecniche di forging dei pacchetti sono documentate ufficialmente, in quanto sono disponibili per usi legali, o quanto meno, non maliziosi, come la gestione di protocolli non implementati direttamente nel kernel.

RAW Sockets (2)

Poche righe di codice C sono necessarie per creare ad esempio un segmento TCP:

```
...
char buffer[TCP_BUFF];
struct tcphdr *tcp;
...
tcp=(struct tcphdr *)(buffer);
tcp->source=source;
tcp->dest=port;
tcp->seq=seq;
tcp->doff=5;
tcp->syn=1;
tcp->window=htons(TCP_WIN);
...
```

RAW Sockets (3)

L'attaccante quindi può inviare datagrammi IP completi e validi all'interno della rete al posto del kernel. Questo permette ad esempio di modificare l'indirizzo IP sorgente, specificare il valore di SEQ e ACKN, inizializzare i bit di controllo di TCP (ovvero SYN, ACK, PSH, URG, FIN, RST).

Unitamente alle informazioni derivanti dallo sniffing e dall'analisi e predizione dell'ISN, può quindi creare, modificare o interrompere connesioni TCP.

RAW Sockets (4)

Una possibile difesa consiste nell'impedire ad un attaccante di inviare in rete datagrammi con indirizzi IP contraffatti. Due gli approcci:

- impedire dal singolo host di inviare datagrammi con indirizzi IP non associati alle interfacce del sistema

- impedire ai clienti di un carrier di inviare datagrammi con indirizzi IP non associati ai net blocks del carrier stesso

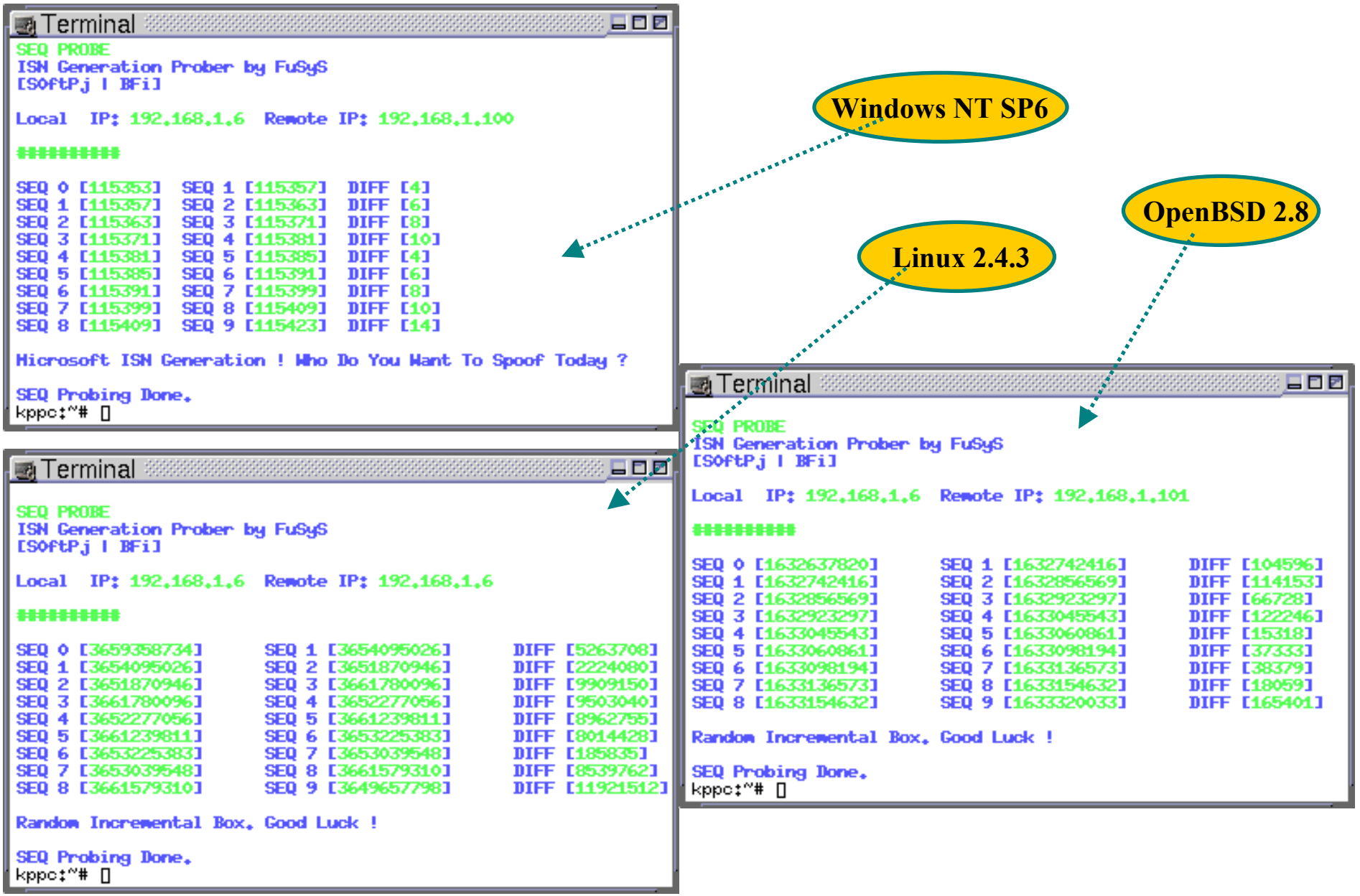
Analisi e Predizione dell'ISN

Mediante la visualizzazione del traffico di rete, è possibile per l'attaccante utilizzare normali richieste di connessione TCP come sonde per ottenere i valori dell'ISN che il generatore del kernel del sistema bersaglio ha computato.

A seconda dei generatori, è possibile predire o quanto meno restringere il range di possibili valori del prossimo ISN di risposta ad una ulteriore richiesta di connessione.

Questa predizione, unitamente alla possibilità di creare datagrammi mediante socket di tipo RAW, permette di forgiare connessioni TCP con indirizzo IP sorgente modificato.

Analisi e Predizione dell'ISN (2)



Analisi e Predizione dell'ISN (3)

Come dimostrato nel documento di **Bindview**, le possibilità di un attaccante sono discrete anche nel caso di generatori pseudocasuali. Questo a patto che l'attaccante abbia le risorse necessarie in termini di banda per poter tentare tutto il range di ISN predetti.

Per generatori conformi all'RFC793, RFC1948 semplice e per i generatori di sistemi Windows 9x e vecchie release della maggior parte di ogni sistema, lo sforzo necessario alla predizione risulta invece **quasi nullo**.

Attacchi al TCP

TCP Spoofing

TCP Hijacking

Man in the Middle

TCP Mangling

TCP Spoofing

- permette di forgiare connessioni TCP con indirizzo IP contraffatto
- permette di evitare autenticazioni basate sul solo indirizzo IP
- permette di rimanere virtualmente irrintracciabili

TCP Spoofing (2)

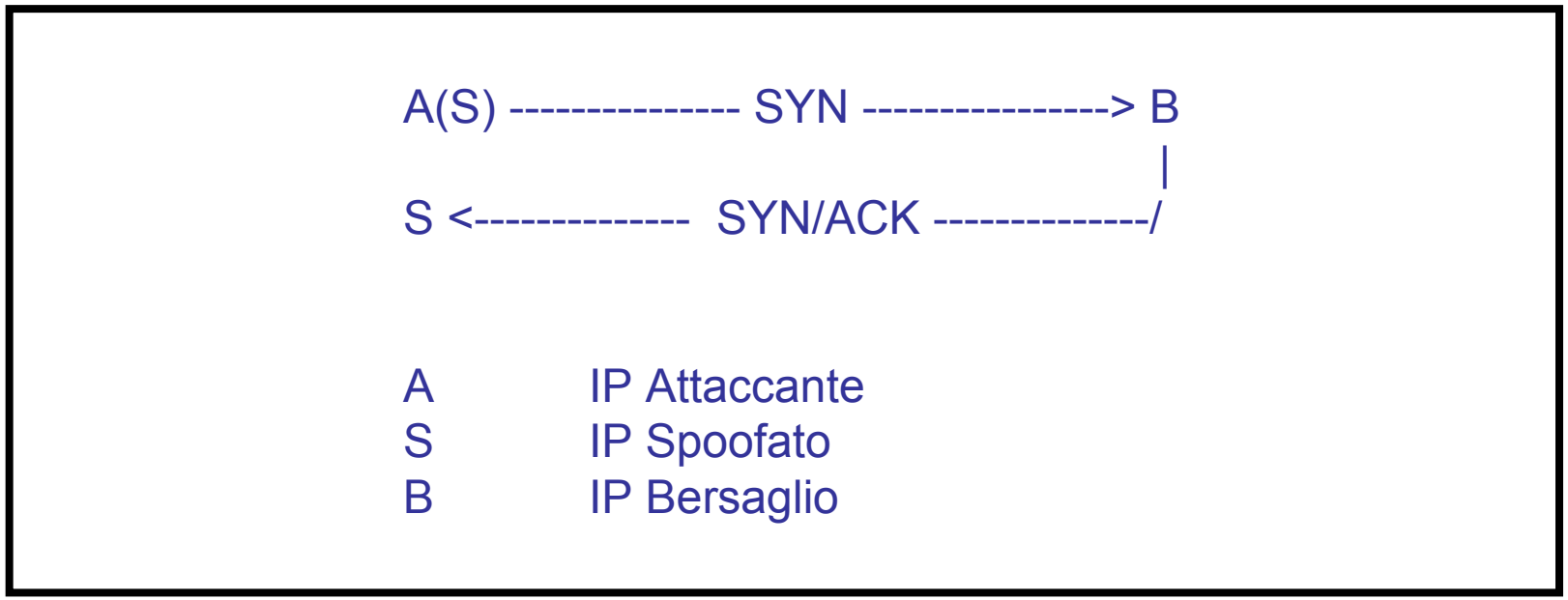
L'attaccante sfrutta la possibilità di predire l'ISN del server per rispondere in maniera valida alla generazione del SEQ senza visualizzare il segmento TCP del server in risposta ad una richiesta con bit SYN.

I problemi per l'attaccante sono dovuti a:

- la sua **cecità**
- le risposte del sistema cui appartiene l'interfaccia con l'indirizzo usato

TCP Spoofing (3)

La **cecità** dell'attaccante è ovvia. Infatti il sistema bersaglio invierà il segmento con bit SYN e ACK all'indirizzo IP usato dall'attaccante e non alla interfaccia del sistema usato:



TCP Spoofing (4)

Il sistema in possesso dell'indirizzo S risponderà, da RFC, con un segmento contenente il bit RST per abortire una connessione non richiesta. Questo segmento, se arrivato prima del completamento dell'attacco, frustrerà il tentativo dell'attaccante.

Questo problema non è presente se l'attaccante utilizza un indirizzo S non esistente, o se l'interfaccia associata a S è down, in quanto non si avrà la risposta di RST.

TCP Spoofing (5)

L'attaccante risolve il problema della cecità mediante la predizione dell'ISN, inviando il segmento contenente l'ACK senza aspettare il SYN/ACK del server.

Risolve il problema del RST del sistema impersonato disabilitando, saturando, manipolando le comunicazioni o le risorse del sistema di cui spoofa l'indirizzo (esaurimento TCB, reboot in seguito a DoS, alterazione routing, ...)

TCP Spoofing (6)

La sola generazione pseudocasuale non sembra sufficiente; nelle parole di **Zalewski**,

"***The general conclusions can seem a little scary.***“

Un approccio valido può essere l'utilizzo di metodi analoghi a quello dei **Syn Cookies** di **Bernstein** e **Shenk**. Autenticazione crittografica forte mediante sistemi di challenge/reply.

L'utilizzo di tecniche di incapsulamento crittografico per l'autenticazione, come **IPsec**, non è adatto al normale uso di protocolli come HTTP e SMTP su larga scala da parte degli utenti dial-up.

TCP Hijacking

- permette il dirottamento di sessioni TCP
- permette di bypassare sistemi di autenticazione come One-Time Password, Kerberos, autenticazioni generiche mediante token

TCP Hijacking (2)

Il caso classico: furto di sessioni terminale remote.

Dalla presentazione del paper di **Joncheray** nel 1995, sono usciti vari tool in grado di operare semplice hijack di sessioni TCP (**Juggernaut**, **xTHOT**, **Hunt**, ...)

TCP Hijacking (3)

L'attaccante dev'essere in grado di sniffare il traffico di rete tra le due parti. Questo permette visione e controllo dei numeri di sequenza.

Passo obbligatorio: desincronizzare la comunicazione:

- a **metà del 3-way**, inviando per conto del client un **RST** dopo il **SYN/ACK** del server e prima dell'**ACK finale**. Questo ACK del client non sarà accettato e questi si troverà in **stato ESTABLISHED ma desincronizzato**. Nel mentre, l'attaccante invia un nuovo SYN per rispondere poi con un ACK. Questo metodo non consente di aggirare autenticazioni, ma permette il furto delle connessioni

- **inviando dati dopo lo stabilirsi della connessione**. Questo porta avanti l'**ACK** del server, rendendo **non validi i successivi SEQ del client** legale. L'attaccante invece è a conoscenza dello stato degli ACK del server e può quindi portare avanti una connessione già iniziata. Il client rimane desincronizzato ed immobilizzato.

TCP Hijacking (4)

Per terminare la connessione, l'attaccante invia segmenti contenenti bit **RST**.

Potendo mantenere lo stato dei numeri di sequenza del server mediante sniffing, può far convalidare il suo **RST** senza problemi.

TCP Hijacking (5)

Un semplice ed effettivo metodo contro questo semplice attacco consiste nell'uso di **crittografia** sul payload dei segmenti TCP. Questo impedisce all'attaccante di portare avanti la connessione, non possedendo le necessarie informazioni per l'autenticazione dei dati.

La crittografia non protegge però dal **TCP Mangling**

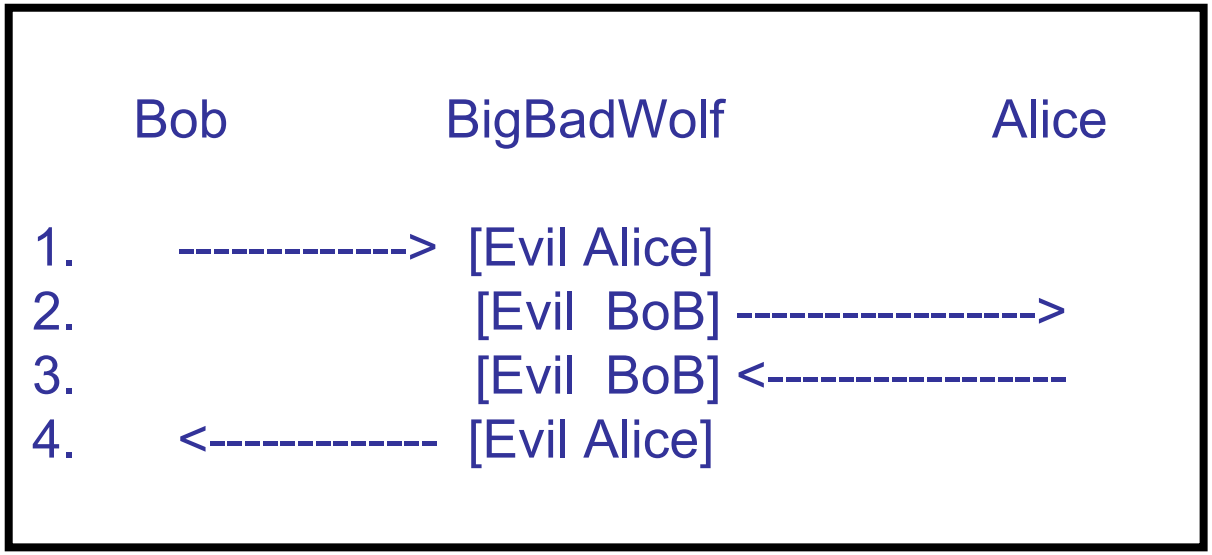
Man in the Middle

- permette di aggirare autenticazioni e protezioni crittografiche
- permette di spacciarsi per terze parti considerate fidate e/o sicure

Man in the Middle (2)

Esistono strumenti scaricabili, come **Ettercap** e **DSniff** che permettono di operare questo tipo di attacco.

Nel caso più semplice, **Bob** e **Alice** intendono comunicare mediante TCP. Attraverso lo sniffing, il re-routing, corruzioni del DNS o altro ancora, **BigBadWolf** è in grado di inserirsi nel normale flusso prima che questo abbia luogo.



Man in the Middle (3)

In questo caso l'uso dei **socket RAW** e dello **sniffing** servono per mantenere **Bob** e **Alice** nell'illusione di comunicare tra loro, quando invece comunicano con il solo **BigBadWolf**.

BigBadWolf comunicherà con **Bob** spacciandosi per **Alice** e viceversa.

Man in the Middle (4)

Come aggirare autenticazioni, ad esempio, mediante **SSHv1** e **3DES** ?

Dopo aver corrotto le cache **ARP** l'attaccante instraderà tutto il traffico attraverso se stesso. A questo punto è necessario sostituire la chiave pubblica di **Alice** con una generata appositamente e inviarla a **Bob** per conto del vero demone **SSH**.

Bob invia la chiave di sessione cifrata con quella contraffatta, decifrabile dall'attaccante, che può estrarre la reale chiave di sessione **3DES**. Questi invia ad **Alice** una nuova versione usando la chiave legale del demone **SSH**. La connessione ha luogo, ma ora **BigBadWolf** possiede la chiave di sessione.

Man in the Middle (5)

L'unica difesa auspicabile consiste nell'impedire la manipolazione degli header TCP. **IPSec** in modalità di trasporto mediante AH, Authentication Header, permetterebbe di proteggere gli header TCP dalla manipolazione.

Purtroppo IPSec non è utilizzabile su larga scala al momento. Molteplici sarebbero i problemi per gestire chiavi per ogni parco utenti dial-up verso ogni possibile server remoto. Per questo viene indicato come strumento adatto a VPN.

Comunque **IPSec** è stato oggetto di critiche e problemi. Un documento che analizza questi problemi è stato presentato da **Ferguson** e **Schneier**, dal titolo "**A Cryptographic Evaluation of IPsec**". Nelle parole degli autori:

"***IPsec was a great disappointment to us.***"

TCP Mangling

- permette di interrompere connessioni TCP
- permette di operare Denial of Service
- permette di facilitare attacchi di tipo Man in the Middle

TCP Mangling (2)

L'attaccante può inserire segmenti TCP con bit **RST** in modo da interrompere la normale comunicazione. Per far questo deve solo essere in grado di visualizzare il traffico e monitorare lo stato dei numeri di sequenza.

Un semplice segmento con indirizzo IP del server inviato subito dopo una sua risposta, che contenga un **SEQ** uguale a quello appena sniffato, ma aumentato della lunghezza del segmento, ed un **ACKN** identico, in quanto il client non avra' ancora inviato suoi dati, con bit **RST** abortira' la connessione.

Il client riceverà un laconico e semplice

"Connection reset by peer."

TCP Mangling (3)

Interrompendo l'inizializzazione di connessioni **SSH**, il client riceverà la proposta di passare ad una connessione non sicura. Osservazioni dimostrano che spesso in ambienti multi-utente è frequente il comportamento non curante da parte degli utenti.

Uno sniffer modificato per interrompere sessioni **SSH** in una **LAN** potrebbe registrare password da sessioni telnet o rlogin usate come ripiego.

L'effetto più banale consiste in un **DoS** per l'impossibilità di portare a termine connessioni di qualunque tipo, se l'attaccante vuole.

TCP Mangling (4)

Interrompendo tentativi di connessione con protocolli come SSH, HTTPS, SMTP, POP3 ed altri, è possibile facilitare un approccio tipo **Man in the Middle**, dando tempo all'attaccante di sincronizzare l'impersonificazione.

Spesso tali errori non sono valutati con preoccupazione dagli utenti, ma imputati ai "*soliti disguidi e rallentamenti della rete*".

Difese ?!

Non esiste alcun metodo omnicomprensivo per esaurire il rischio di sovversioni di un protocollo come TCP. Necessario è invece adeguare le operazioni di **network** e **host security** per evitare che l'attaccante abbia facile accesso a risorse indispensabili per questi attacchi. Ad esempio:

- migliore gestione degli IDS
- hardening di ogni macchina
- policy di sicurezza locale

Considerazioni

TCP, così come architettato ed implementato, **non** andrebbe considerato un protocollo sicuro.

L'aggiunta di sessioni crittografiche migliora sensibilmente la situazione, ma non è in grado di risolvere il problema nel momento in cui l'attaccante possa avere controllo diretto della LAN, di hop intermedi, delle tabelle di risoluzione dei nomi o degli instradamenti.

La facilità con cui è possibile per un attaccante monitorare e manipolare il traffico di rete è disarmante. Nel caso di attaccanti non evoluti, è fonte di preoccupazione l'enorme disponibilità di ottimo codice di attacco presente in rete.

Questo va combattuto non con least (o **NULL**) disclosure, bensì con una maggiore attenzione alla implementazione di sistemi di difesa.

Riferimenti

Anti Antisniff Patch pubblicato su BFi9
http://www.s0ftpj.org/bfi/

Progetto Onosendai pubblicato su BFi4 e BFi6
http://www.s0ftpj.org/bfi/

DDos Pet Nemesis: spoofing detection pubblicato su BFi8
http://www.s0ftpj.org/bfi/

Italian Security Mailing List
http://www.sikurezza.org/

DSniff http://packetstorm.mirror.widexs.nl/sniffers/dsniff/

Ettercap http://ettercap.sourceforge.net/

AntiSniff http://packetstorm.mirror.widexs.nl/sniffers/antisniff/

E-Zine Phrack http://www.phrack.org/

Documenti

RFC791, 793, 1948

CERT® CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks

"A weakness in the 4.2BSD UNIX TCP/IP Software",
Robert T.Morris, 25/2/1985

"Security Problems in the TCP/IP Protocol Suite"
S.M.Bellovin, aprile 1989

"A Cryptographic Evaluation of IPsec" N.Ferguson e B.Schneier

"A Simple Active Attack Against TCP", L. Joncheray, 1995

"The problems With Random Increments", T.N.Newsham

"Strange Attractors and TCP/IP Sequence Number Analysis"
M.Zalewski

SynCookies http://cr.yp.to/syncookies.html

Italian Black Hats Association

Sovversioni del protocollo TCP/IP: attacchi e contromisure

24 gennaio 2002, Infosecurity Italia

Relatore:

Matteo Falsetti fusys@blackhats.it

BlackHats.it

General Infos: info@blackhats.it