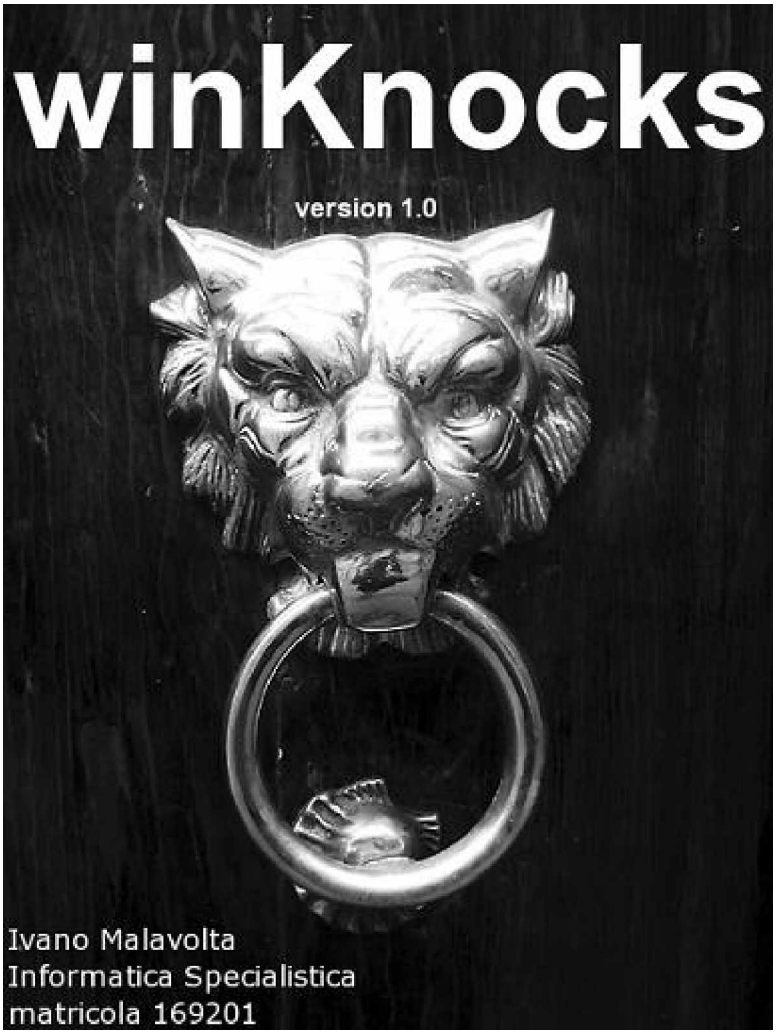


Indice

1 Introduzione

Questo capitolo contiene una descrizione riassuntiva del tool winKnocks e presenta i concetti base del port knocking. winKnocks è un port knocking tool con sequenze criptate per sistemi Windows. Le knock sequences sono definite attraverso file XML. L'utente può specificare il numero di pacchetti di ogni knock sequence, payload e tutti i campi dell'header di ogni pacchetto. Il tool è dotato di un logger che tiene traccia del traffico potenzialmente pericoloso, genera smoke packets (che verranno descritti in seguito) e contiene un meccanismo che lo rende immune ai cosiddetti "replay attacks". Una descrizione più dettagliata delle varie funzionalità e delle tecniche usate verrà esposta nei capitoli successivi.

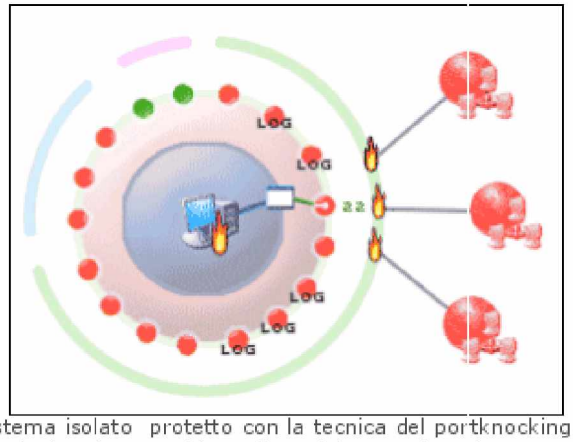
1.1 Cos'è il PortKnocking

La tecnica del portknocking è un sistema per comunicare con un computer in rete che non richiede alcuna porta aperta per poter funzionare. Non e' infatti necessario mantenere i corrispondenti servizi costantemente in ascolto e pronti a ricevere connessioni. Questa sua caratteristica costituisce indubbiamente un grande vantaggio: finché una porta rimane aperta, il servizio applicativo ad essa associato e' in grado di accettare connessioni, restando quindi vulnerabile a diversi attacchi. Prima che una connessione ad uno specifico servizio possa essere stabilita viene usata una speciale sequenza di pacchetti di rete, ovvero una serie di tentativi di connessione verso delle porte di rete chiuse come meccanismo di autenticazione del richiedente del servizio. Il richiedente "bussa alle porte" e si fa riconoscere prima di poter accedere, da qui il nome "portknocking". La sequenza di pacchetti utilizzata per autenticarsi prende il nome di sequenza di knock. Sfruttando questa tecnica e' quindi possibile manipolare dinamicamente le regole di filtraggio, aprendo una o più porte solo quando necessario, una volta autenticatosi il richiedente. Naturalmente possono esistere numerose sequenze, associate ad azioni differenti. Ad esempio si possono prevedere sequenze di knock diverse per l'accesso ai singoli servizi applicativi.

Viene ora illustrato un esempio pratico dell'utilizzo di questa tecnica, per gestire l'accesso al servizio SSH su una macchina Linux che utilizzi iptables. Il server sul quale il servizio e' in esecuzione presenta quindi tutte le porte di rete chiuse, ad eccezione delle cosiddette porte dinamiche, per mezzo della seguente regola di filtraggio:

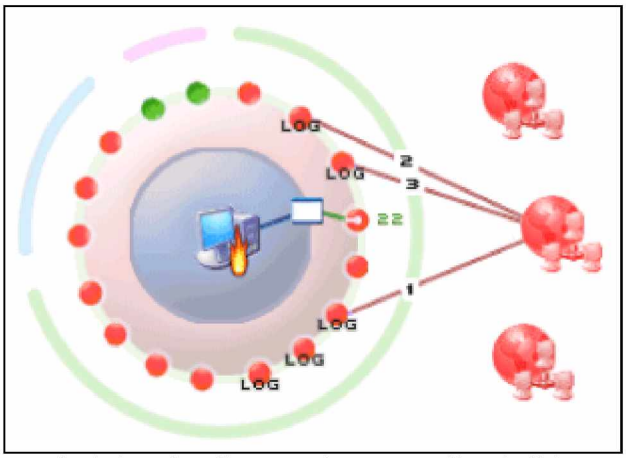
```
iptables -P INPUT DROP
```

La scelta di utilizzare l'azione DROP anziché REJECT, e' giustificata dal fatto che essa si limita a scartare il pacchetto arrivato, senza inviare alcun pacchetto di errore ICMP ai richiedenti il servizio. Viene così reso indistinguibile il rifiuto del pacchetto dall'assenza del server.



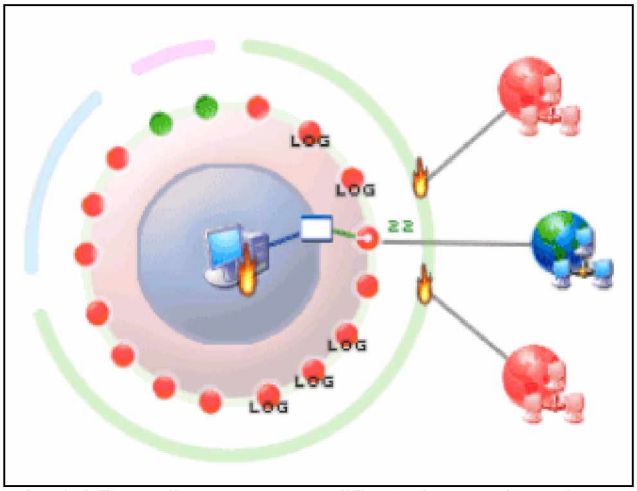
Un sistema isolato protetto con la tecnica del portknocking. Tutte le porte sono chiuse, inclusa la 22, sebbene il servizio SSH sia attivo.

Un utente fidato per potere accedere al servizio, deve iniziare una serie di connessioni verso diverse porte secondo un certo ordine. La particolare sequenza di knock ovviamente è un segreto noto esclusivamente agli utenti che devono essere abilitati all'utilizzo del servizio applicativo, ed al software lato server responsabile del riconoscimento delle sequenze stesse.



Un utente legittimo localizzato ad uno specifico indirizzo IP, tenta una serie di connessioni verso delle porte chiuse secondo un preciso ordine.

Il portknocking server, riconoscendo una particolare sequenza, può modificare le regole di filtraggio, ad esempio aggiungere una regola per mezzo di iptables, che consenta l'accesso al servizio SSH all'indirizzo IP del richiedente, come illustrato dall'immagine successiva :



Le regole del firewall vengono modificate in seguito ad una corretta sequenza di knock, in modo da permettere al richiedente di connettersi alla porta 22.

Per esempio un utente può cercare di connettersi in sequenza alle porte 24, 20 e 21. Dal suo punto di vista i tentativi di connessione non hanno avuto successo, come se la macchina di destinazione non esistesse. I pacchetti della sequenza di knock vengono infatti scartati dal firewall in maniera "silenziosa", senza cioè ritornare al richiedente alcun messaggio d'errore.

Nel frattempo però, il portknocking server interpreta la sequenza di knock ricevuta e nel caso essa sia giudicata valida, può eseguire l'azione associata. Nel caso illustrato si potrebbe aprire la porta 22 all'indirizzo IP del richiedente, con il seguente comando :

```
iptables -A INPUT -s $IP -p tcp --dport 22 -j ACCEPT
```

Sfruttando il medesimo meccanismo, una volta che l'utente abbia terminato le sue operazioni, e' possibile specificare un seconda sequenza per chiudere la porta precedentemente aperta tramite il comando :

```
iptables -D INPUT -s $IP -p tcp --dport 22 -j ACCEPT
```

Per l'implementazione di un simile sistema esistono varie possibilità e criteri su cui basarsi, che verranno discusse dettagliatamente nei capitoli successivi.
Volendo però riassumere, le metodologie realizzative si riducono alle seguenti tre:

1. Monitoraggio dei log di sistema;
2. Analisi real-time del traffico di rete al livello Datalink(winKnocks è stato implementato in questo modo);
3. Riutilizzo di meccanismi di Intrusion Detection quali snort oppure hogwash.

In conclusione, la tecnica del portknocking, quindi, e' una sorta di protezione adatta ai casi in cui ci sono utenti che richiedono un accesso continuo a dei servizi che non devono essere pubblici come invece sono http o SMTP, mantenendo le porte di rete chiuse al traffico pubblico, ed aprendole o chiudendole in maniera flessibile ai richiedenti che si siano autenticati con una sequenza di knock corretta.

2 winKnocks

Il meccanismo più usato dai portKnocking tool è quello di analizzare i files di log del firewall del sistema; questa soluzione però ha numerose limitazioni, come il grande carico computazionale della componente server (che periodicamente deve leggere tutti i file di log del firewall) e il fatto che le knock sequences non sono riconosciute in modo tempestivo. Visto le pesanti limitazioni di questo modello è necessaria una soluzione meno costosa dal punto di vista computazionale e che possa lavorare più a stretto contatto con il firewall, riconoscendo i pacchetti al momento del loro arrivo invece che leggendo la storia recente dai log.
winKnocks è un port knocking tool con sequenze criptate per sistemi Windows. Come già detto in precedenza, esso si basa sulla cattura di pacchetti a livello DataLink; tale meccanismo è basato sulla libreria Jpcap, che a sua volta fa utilizzo di Winpcap.
La prima peculiarità di winKnocks è che le regole di filtraggio vengono applicate direttamente al firewall di Windows. In questo modo l'utente non è costretto ad utilizzare per forza un firewall ad-hoc per il portknocking. Le regole di filtraggio vengono generate automaticamente dal tool in base alle knock sequences riconosciute e vengono applicate attraverso l'utility di Windows "netsh".
L'intero progetto può essere diviso in due parti: la componente *Knocker* e quella *Listener*. Entrambe sono scritte in Java ed hanno l'interfaccia grafica Java-Swing. I dettagli implementativi si possono trovare nel prossimo capitolo(2.1). Knocker e Listener condividono files XML contenenti la descrizione delle knock sequences definite dall'utente. Quest'ultimo può definire: il numero dei pacchetti della knock sequenze, il payload e i vari campi dell'header di ogni pacchetto (che può essere di tre tipi: UDP, TCP, ICMP). Durante l'invio di una sequenza l'utente può definire un "urgent script" da eseguire server-side. Il listener può permettere o no l'esecuzione di tali script. Ad ogni knock sequence possono essere associati tre tipi di azioni: apertura di una porta, chiusura di una porta, esecuzione di uno script bash. Nel capitolo 2.2 verranno descritte nel dettaglio le numerose varianti di cui può godere una knock sequence. Il payload dei pacchetti è criptato utilizzando l'algoritmo DES ed una password segreta condivisa da Knocker e Listener. Il Listener

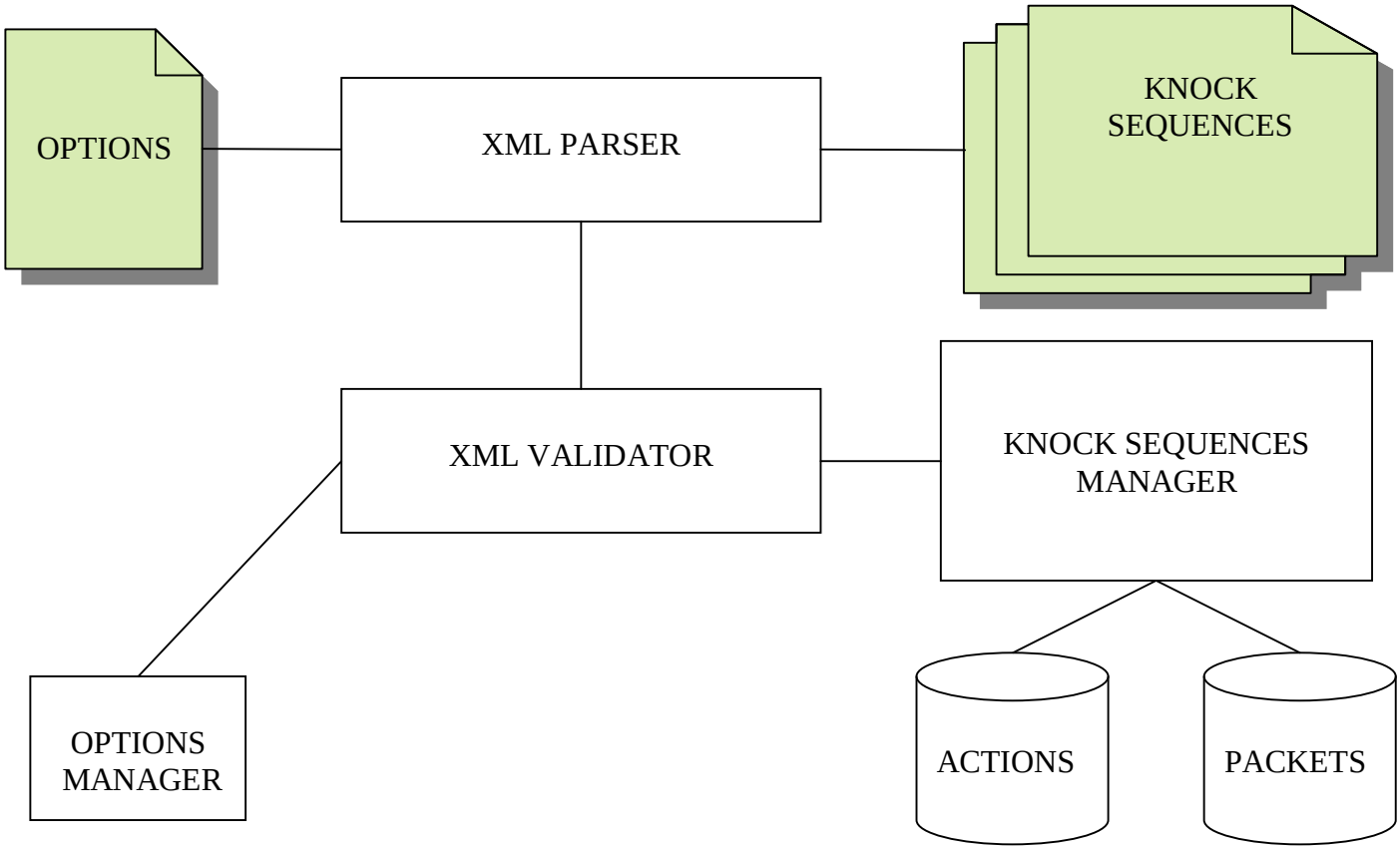
inoltre ha capacità di logging. Il knocker, oltre ai pacchetti della sequenza, invia anche *smoke packets,* con lo scopo di rendere difficile un eventuale sniffing del traffico diretto verso il Listener.
Quest'ultimo è in grado di distinguere i pacchetti delle sequenze dagli smoke packets attraverso due meccanismi, che verranno descritti nel capitolo 2.4. Durante lo sviluppo del tool sono state prese in considerazione numerose soluzioni per ovviare al problema dei *replay attacks,* la soluzione finale è presentata nel capitolo 2.5. Infine, in caso di tentativi di intrusione, i pacchetti ricevuti e indirizzo IP del mittente sono loggati.

2.1 Descrizione ad alto livello dell'architettura del tool

WinKnocks è stato scritto in Java ed ha un'interfaccia grafica Swing. Le motivazioni per cui è stato scelto questo linguaggio di programmazione sono:

- le knock sequences sono definite attraverso files XML; è noto che Java sia uno dei linguaggi che si interfacciano meglio con XML.
- Esistono numerosi parsers e schema processors XML molto potenti con API Java.
- Facilità di creazione e flessibilità della GUI.
- Esiste la libreria Jpcap, che rende l'accesso alla rete da parte della componente Listener molto semplice e "customizzabile".
- Il tool deve essere particolarmente resistente agli attacchi di tipo Denial of Service e non deve mai stare "down"; i meccanismi di sicurezza Java lo rendono resistente a questi tipi di vulnerabilità.

Il motivo per cui il sistema non deve mai stare "down" è che esso manipola dinamicamente le regole di filtraggio del firewall. Basti pensare alla situazione in cui il processo del portknocking server dovesse morire inaspettatamente o non funzionare correttamente: non sarebbe più possibile interpretare le sequenze di knock e come diretta conseguenza diventerebbe impossibile connettersi all'host protetto.
Continuando, è stato scelto di definire le sequenze di knock attraverso files XML per la facilità da parte dell'utente di creare tali files, per il fatto che la struttura dei files (che verrà descritta in seguito) è molto intuitiva, per la portabilità di XML; il parsing di files XML inoltre è molto facile e potente attraverso le librerie adatte. Con l'utilizzo di XML schema inoltre è impossibile creare knock sequences sintatticamente o semanticamente scorrette, quindi non è stato necessario fare un filtraggio dei dati immessi dall'utente.
L'intero progetto è composto da due componenti indipendenti: Knocker e Listener. Tuttavia essi hanno un nucleo interno comune; la figura nella pagina seguente illustra le componenti che fanno parte del nucleo interno.

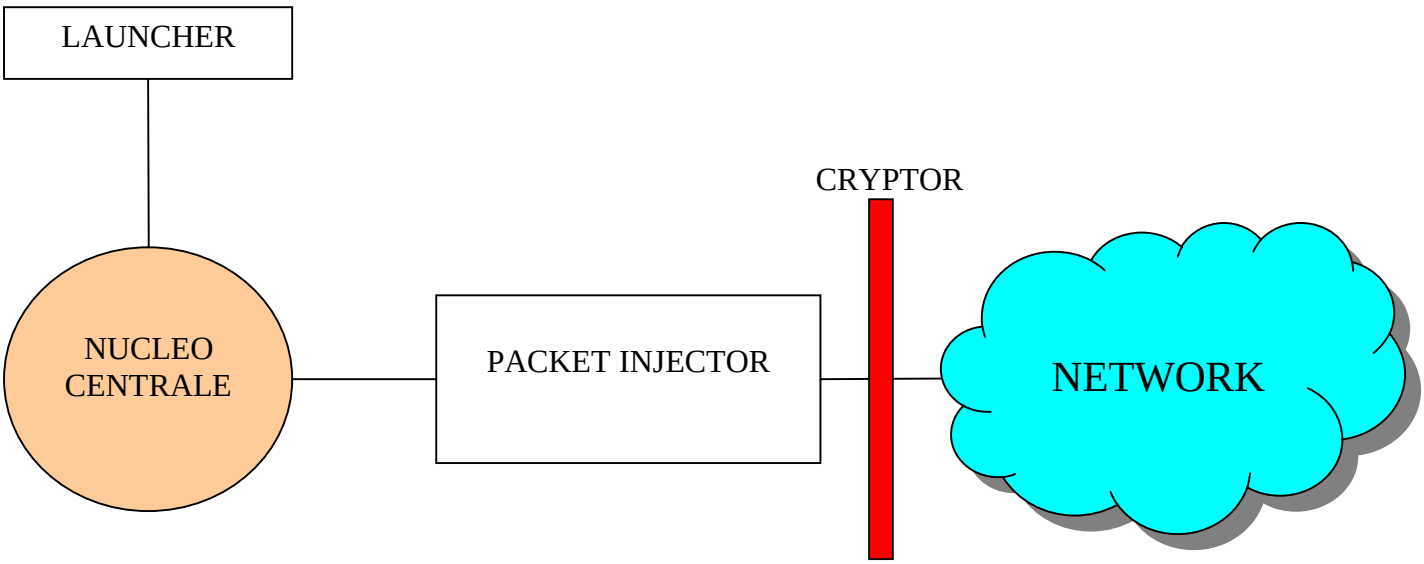


Le figure verdi (*OPTIONS* e *KNOCK SEQUENCES*) rappresentano i files XML delle opzioni e delle knock sequences.
XML PARSER è una componente che legge i file XML e verifica che siano sintatticamente corretti. *XML VALIDATOR* invece analizza i dati immagazzinati da *XML PARSER* e verifica che essi siano anche semanticamente corretti. Alcuni controlli semantici che esegue questa componente sono:

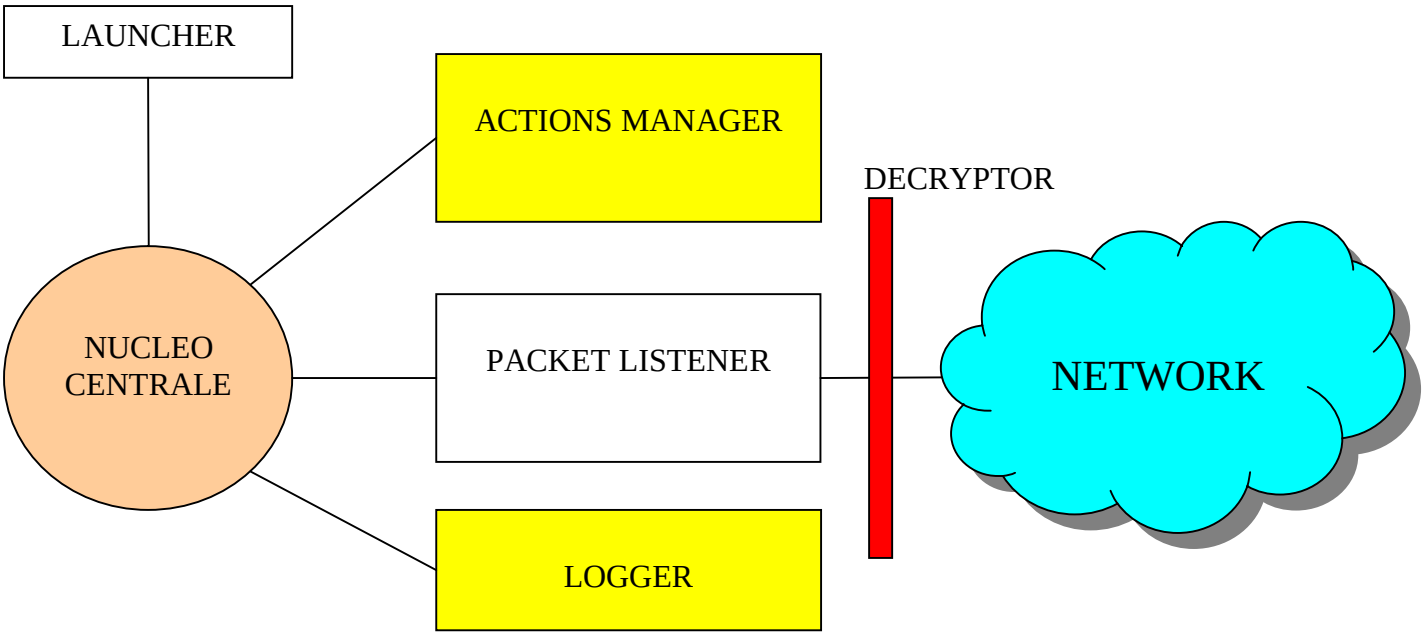
- verificare che i numeri di porta all'interno dei tags OpenPort o ClosePort siano degli interi compresi tra 0 e 65535;
- verificare che ogni knock sequence abbia un id unico all'interno del sistema;
- verificare che i valori che l'utente ha inserito all'interno dei flags TCP siano dei booleani
- e cosi via...

Continuando, OPTIONS MANAGER è la componente che tiene traccia delle opzioni che l'utente ha settato e che verifica che esse siano corrette; per esempio controlla che l'utente abbia inserito un identificativo corretto dell'interfaccia su cui immettere le knock sequences,ecc.
KNOCK SEQUENCES MANAGER invece è la componente che tiene traccia delle knock sequences definite dall'utente; ovviamente verifica che esse siano ben formate.

Tutta questa struttura è il nucleo interno sia della componente Knocker sia di quella Listener. Questo nucleo centrale fornisce tutte le informazioni necessarie di cui il sistema ha bisogno per inviare e ricevere knock sequences.
Qui di seguito sono descritte separatamente le architetture di Knocker e Listener. La figura nella pagina seguente illustra l'architettura della componente Knocker.



LAUNCHER è la componente di start-up e serve per istanziare il nucleo centrale; inoltre avvia tutti i controlli sui files XML definiti dall'utente e lancia l'interfaccia grafica. *PACKET INJECTOR* invece riceve le richieste di invio di pacchetti e invia questi ultimi sulla rete. Questa componente verrà analizzata nel dettaglio nel capitolo 3.2. *CRYPTOR*, infine, serve per criptare il payload di ogni pacchetto uscente dalla componente *PACKET INJECTOR*.
La figura seguente illustra l'architettura della componente Listener.



LAUNCHER è la componente di start-up e serve per istanziare il nucleo centrale, avvia tutti i controlli sui files XML definiti dall'utente e lancia l'interfaccia grafica. *PACKET LISTENER* è la

componente di packet filtering che analizza i pacchetti entranti a livello IP; essa riceve i pacchetti delle knock sequences e li analizza (in base alle informazioni contenute nel *NUCLEO CENTRALE*). *ACTIONS MANAGER* serve per aprire/chiudere le porte del firewall o eseguire scripts in base alle knock sequences ricevute dal *PACKET LISTENER*. *LOGGER* invece è la componente che esegue il logging di tutte le attività di winKnocks. Maggiori informazioni sulla fase di logging si trovano nel capitolo 2.6. La componente *DECRYPTOR* infine serve per filtrare i pacchetti entranti e decriptare i rispettivi payload utilizzando la chiave segreta memorizzata nel *NUCLEO CENTRALE*.

2.2 Le knock sequences

Ad ogni knock sequence corrisponde un file XML contenente alcune impostazioni, una lista di azioni(paragrafo 2.2.1) e una lista di pacchetti(paragrafo 2.2.2). Nella creazione di una knock sequence l'utente può definire:

- *id*: stringa che identifica univocamente la sequenza; durante la creazione il tool genera automaticamente un id non ancora utilizzato.
- *Description*: descrizione riassuntiva della sequenza; può essere utile nel caso in cui l'utente ha definito molte knock sequences, quindi in questo campo potrebbe mettere una descrizione di ciò che fa la sequenza.
- *SmokePackets*: questo campo è composto da due sotto-campi: min e max. Essi contengono rispettivamente il numero minimo e massimo di smoke packets da inviare insieme alla knock sequence. Questa impostazione fornisce grande flessibilità; ad esempio infatti per knock sequences particolarmente "sensibili" l'utente potrebbe definire un alto numero di smoke packets; se l'utente non volesse generare un grande overhead di traffico potrebbe settare a zero tali valori.
- *MaxFakePayload*: questo campo contiene il numero massimo di caratteri random da aggiungere alla fine del payload di ogni pacchetto. Se non ci fosse questa caratteristica, la lunghezza dei payload dei pacchetti potrebbe variare pochissimo, rendendo le knock sequences facilmente riconoscibili da un eventuale sniffer. Quindi per ogni pacchetto (anche per gli smoke packets), il tool genera un numero random *x* compreso tra 0 e *MaxFakePayload* e in seguito aggiunge al payload *x* caratteri random.

Nei prossimi due paragrafi sono descritti nel dettaglio le azioni e i pacchetti che si possono associare ad una knock sequence.

2.2.1 Tipi di azione

Nel file XML è presente una lista di Actions; esse corrispondono alle azioni che il listener dovrà compiere dopo aver ricevuto correttamente una knock sequence. Le azioni verranno eseguite nell'ordine in cui sono state definite nel file XML. WinKnocks da la possibilità di definire tre tipi di azioni: OpenPort, ClosePort, ExecuteScript. Esse corrispondono rispettivamente all'apertura, chiusura di una porta (definendole applicando dinamicamente una nuova regola del firewall di Windows), e all'esecuzione di uno script bash. OpenPort è una delle azioni più interessanti anche per il numero di impostazioni che il tool permette di definire, che sono:

- *PortNumber*: ovviamente, il numero di porta da aprire;
- *Exclusive*: questo è un campo di tipo booleano. Se è false la porta è aperta a tutti, altrimenti essa risulta aperta solo all'IP che ha inviato la knock sequence. Questa è una peculiarità di winKnocks; infatti analizzando numerosi altri tool di portknocking ho notato che,

nonostante essi abbiano questa capacità, non permettono all'utente di scegliere se la porta è aperta a tutti oppure no;
- *Wait*: questo campo è comune a tutte le azioni; permette di definire un tempo di attesa dopo il quale verrà eseguita l'azione associata. Per esempio, un OpenPort con il campo Wait = 30 indica il fatto che solo dopo 30 secondi la ricezione della knock sequenze si aprirà la porta in questione;
- *TimeOut*: questo campo è associato solo a OpenPort e ClosePort; esso consente di definire un timeout, dopo il quale la porta si richiude/riapre automaticamente.

L'azione ClosePort è simile a OpenPort, solo che non è possibile utilizzare Exclusive. L'azione ExecuteScript infine ha solo due campi:
- *Script*: il path assoluto dello script da eseguire;
- *Wait*: già decritto in precedenza.

2.2.2 Tipi di pacchetto

Nel file XML è presente una lista di pacchetti, che corrispondono ai pacchetti che il listener deve ricevere per eseguire le Actions descritte sopra. Da notare che il listener deve ricevere i pacchetti nello stesso ordine in cui sono descritti nel file XML. Questa è una limitazione comune a tutti i tool di portKnocking, in Internet infatti l'ordine di arrivo dei pacchetti non è garantito.
Come detto in precedenza winKnocks permette l'invio di tre tipi di pacchetti: UDP, TCP e ICMP. Per ogni singolo pacchetto l'utente può settare i tutti i campi dell'header (appendice A) e il contenuto del payload. Maggiori informazioni su come viene gestito il payload dei pacchetti verranno fornite nel prossimo paragrafo.
La scelta di permettere l'utilizzo di questi tre protocolli è stata dettata dai seguenti fattori:
- semplificare l'utilizzo del tool agli utenti che non hanno grandi conoscenze di networking;
- le knock sequences con protocolli "strani" verrebbero facilmente identificate da un eventuale intrusore che stia sniffando il traffico di rete;
- questi sono i protocolli più utilizzati, e quindi si uniformano facilmente al traffico di rete che li circonda, rendendo l'identificazione delle knock sequences molto più difficile.

Infine, io consiglierei di creare knock sequences con soli pacchetti UDP per i seguenti motivi:
- generano poco overhead sul traffico di rete;
- un intrusore che "sniffa" il traffico di rete solitamente non li analizza;
- un intrusore che "sniffa" il traffico di rete si potrebbe insospettire vedendo pacchetti TCP con gli ack number e i sequence number non "allineati", oppure vedendo che connessioni TCP iniziano senza l'hand-shake a tre vie, e cosi via.

Qui di seguito è presente un esempio di file XML relativo ad una knock sequence con un pacchetto TCP, uno UDP ed uno ICMP, con le azioni associate di aprire la porta 23 e il NotePad di Windows.

```
<KnockSequence  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation="KnockSequence.xsd">


        <id>8</id>
        <description>esempio di knock sequence</description>
        <smokePackets>
                <min>10</min>
                <max>50</max>
        </smokePackets>
        <maxFakePayload>100</maxFakePayload>
```

```
        <actions>
                <executeScript>
                        <script>c:/windows/notepad</script>
                </executeScript>
                <openPort>
                        <portNumber>23</portNumber>
                        <exclusive>true</exclusive>
                        <wait>10</wait>
                        <timeout>200</timeout>
                </openPort>
        </actions>

        <packets>
                <TCPpacket>
                        <srcPortNumber>8000</srcPortNumber>
                        <dstPortNumber>9000</dstPortNumber>
                        <sequenceNumber>1</sequenceNumber>
                        <ackNumber>2</ackNumber>
                        <windowSize>50</windowSize>
                        <flags>
                                <ack>0</ack>
                                <syn>1</syn>
                                <fin>0</fin>
                                <push>0</push>
                                <reset>0</reset>
                                <urgent>1</urgent>
                        </flags>
                        <payload>payload</payload>
                </TCPpacket>

                <UDPpacket>
                        <srcPortNumber>7000</srcPortNumber>
                        <dstPortNumber>3000</dstPortNumber>
                        <payload>payload2</payload>
                </UDPpacket>

                <ICMPpacket>
                        <type>8</type>
                        <code>0</code>
                        <data>payload3</data>
                </ICMPpacket>
        </packets>
</KnockSequence>
```

2.3 Payload dei pacchetti e criptografia

La figura sottostante rappresenta la struttura del payload di un pacchetto di una generica knock sequence. Qui di seguito verranno descritti i vari campi. Ovviamente queste informazioni sono

molto sensibili, quindi è stato deciso di criptare il payload del pacchetto attraverso l'algoritmo a chiave privata DES. E' noto che questo algoritmo è insicuro per alcune applicazioni, in quanto la chiave può essere individuata in meno di 24 ore; comunque per l'uso che se ne fa in questo tool è più che sufficiente, poiché ci sono diverse misure di sicurezza che analizzano i valori nei vari campi e individuano facilmente pacchetti non ben-formati. Inoltre, se ci sarà una prossima versione, sarà utilizzato un algoritmo più sicuro, almeno il triple-DES.
Continuando, la chiave per cifrare i payloads dei pacchetti può essere settata utilizzando sia la GUI del Knocker sia quella del Listener, l'importante è che esse siano allineate.

- Payload utente: è la stringa che l'utente può definire nel file XML all'interno del campo Payload relativo ad ogni pacchetto. Può essere utile per identificare ogni singolo pacchetto della knock sequence, o addirittura per trasportare informazione dal Knocker al Listener in modo sicuro.
- ID: questo campo è composto da due sottostringhe: *id* e *randomNumber*, Esse servono per identificare univocamente la knock sequence a cui appartiene il pacchetto. Con questo meccanismo la componente listener è in grado di ricevere knock sequences in parallelo, anche se sono dello stesso tipo e provengono dallo stesso utente. In questo modo si evitano spiacevoli problemi (come quello di ricevere knock sequences e di interpretare male i pacchetti in arrivo), si da la possibilità all'utente di poter creare knock sequences che iniziano con gli stessi pacchetti e si rende la componente Listener molto più veloce.
- TIMESTAMP: è un numero intero. Il knocker riempie questo campo con *X*, dove *X* è la somma del timestamp (in millisecondi) e un numero Random. Lo scopo di questo campo è quello di identificare univocamente ogni singolo pacchetto, in modo da prevenire i replay attacks. Il paragrafo 2.5 contiene una descrizione più dettagliata della questione.
- URGENT SCRIPT: questo campo contiene il path assoluto di uno script da eseguire server-side in "urgent mode", cioè il prima possibile. Lo script da eseguire viene inserito dall'utente prima dell'invio di una knock sequence. Da notare che esso non è relativo ad una knock sequence specifica, ma viene richiesto all'utente appena prima dell'invio di una knock sequence.

La definizione di uno script arbitrario da eseguire server-side potrebbe essere una grossa vulnerabilità all'interno di un sistema (specialmente se il listener è in ascolto sulla rete Internet), per questo motivo ho deciso di lasciare all'utente del Listener la scelta di accettare o no l'esecuzione di scripts in urgent mode. Tale caratteristica può essere settata in qualsiasi momento dalla GUI del listener.

2.4 Smoke packets

Gli smoke packets sono quei pacchetti che il Knocker invia insieme alla knock sequence; in tal modo rende difficile (o impossibile) individuare i pacchetti della knock sequence vera e propria.
Come detto in precedenza, l'intervallo tra il numero massimo e minimo di smoke packets è definito dall'utente; il tool crea un numero random *X* compreso in tale intervallo e invia *X* smoke packets, uniformemente distribuiti tra un pacchetto e l'altro della knock sequence.
A questo punto la questione è: come fa la componente Listener a distinguere gli smoke packets da quelli autentici? La risposta è: attraverso un filtro da applicare alla componente *PACKET LISTENER*, che analizza il traffico di rete entrante.

Tale filtro può essere definito in due modi:

1) Automaticamente; il tool analizza i pacchetti di tutte le knock sequences definite nei files XML e genera in modo dinamico il filtro per il *PACKET LISTENER*, in modo che esso "catturi" solo ed esclusivamente i pacchetti che fanno parte di almeno uno knock sequence e che hanno il payload criptato con la chiave definita dall'utente;
2) Manualmente; è l'utente che crea tale filtro in sintassi TcpDump. Ciò può essere utile nel caso in cui l'utente "non si fidi" del filtro creato automaticamente dal tool, oppure nel caso in cui voglia un filtro ottimizzato al massimo per avere migliori performance.

Comunque la configurazione manuale del filtro è riservata solo ad utenti *advanced*; io consiglio vivamente di scegliere sempre il filtro automatico perché:

1) è creato da un software, quindi non può sbagliare, o ignorare dei pacchetti;
2) è ottimizzato in due steps: la prima fase è eseguita dal tool e la seconda dalla libreria jpcap;
3) è "stretto", è fatto in modo da ricevere solo pacchetti appartenenti a knock sequences.

2.5 Contromisure per i replay attacks

Il contesto dei replay attacks consiste nel fatto che, se il tool non avesse delle contromisure adatte, sarebbe sufficiente "sniffare" il traffico di rete del Listener per determinare la sequenze di knock e poterle quindi riprodurre; generando cosi delle falle nel sistema del Listener.
I replay attacks sono neutralizzati perché, come detto in precedenza, il payload dei pacchetti contiene il campo TIMESTAMP. Il knocker riempie questo campo con *X*, dove *X* è la somma del timestamp (in millisecondi) e un numero Random. In questo modo ogni pacchetto inviato dal Knocker è univocamente identificato. Dall'altro lato, il Listener tiene traccia di tutti i numeri *X* che ha ricevuto e, nel momento in cui ne riceve uno già utilizzato, genera un allarme. In tal caso il pacchetto "incriminato" e l'indirizzo IP che lo ha inviato sono "loggati". Ovviamente i numeri *X* già utilizzati sono memorizzati in una hash-map, in modo da rendere l'accesso a tali valori il più veloce possibile.

2.6 Logging

Come detto in precedenza, la componente Listener ha anche la capacità di logging. Ovviamente non "logga" ogni pacchetto che riceve, ma solo i contesti degni di attenzione, che sono:

- Inizio della ricezione di una knock sequence; in questo caso tiene traccia anche dell'ID della knock sequence.

```
START --- receiving knock sequence with id 0
```

- Fine della ricezione di una knock sequence; tiene traccia dell'ID della knock sequence, dell'esito dell'esecuzione delle azioni associate e degli eventuali urgent scripts eseguiti.

```
END --- knock sequence with id 0 successfully received, all actions have been executed ---
urgent script: none
```

- Ricezione di un pacchetto nell'ordine giusto e con payload ben-formato. In questo caso tiene traccia dell'ordine del pacchetto all'interno della knock sequence e dell'ID di quest'ultima.

```
Sat Aug 12 18:48:56 CEST 2006 --- received well-formed packet; it is the 1st packet of knock sequence with id 0
```

- Ricezione di un pacchetto con payload non criptato o criptato con una chiave diversa. Tiene traccia del pacchetto “incriminato” e dell’indirizzo IP del mittente. Si genera un allarme.

```
INTRUSION DETECTED --- Sat Aug 12 19:10:02 CEST 2006 --- received packet with bad-encrypted payload from IP address 192.168.0.1
   TCPpacket:
        srcPortNumber: 139
        dstPortNumber: 3020
        seqNumber: 1957411988
        ackNumber: 2200705534
        windowSize: 65181
        ACK: true
        SYN: false
        FIN: false
        PUSH: false
        RESET: false
        URGENT: false
        payload: not valid
```

- Ricezione di un pacchetto con payload ben-formato e criptato con la chiave corretta, ma che non è nell’ordine giusto per nessuna knock sequence. Si tiene traccia solo del pacchetto “incriminato” e si genera un “warning”.

```
WARNING --- Sat Aug 12 19:12:09 CEST 2006 --- received packet with well-formed payload, but does not belonging to any knock sequence.
   TCPpacket:
        srcPortNumber: 8000
        dstPortNumber: 9000
        seqNumber: 1
        ackNumber: 2
        windowSize: 50
        ACK: false
        SYN: true
        FIN: false
        PUSH: false
        RESET: false
        URGENT: false
        payload: payload
```

- Ricezione di un pacchetto che era già stato ricevuto in precedenza. Tiene traccia del pacchetto “incriminato” e dell’indirizzo IP del mittente. Si genera un allarme notificando che è avvenuto un replay attack.

```
REPLAY-ATTACK DETECTED --- Sat Aug 12 19:13:34 CEST 2006 --- replay attack detected from IP address 192.168.0.2
   TCPpacket:
```

```
srcPortNumber: 8000
dstPortNumber: 9000
seqNumber: 1
ackNumber: 2
windowSize: 50
ACK: false
SYN: true
FIN: false
PUSH: false
RESET: false
URGENT: false
payload: payload
```

Ogni messaggio di logging è preceduto dal timestamp in cui è successo l'avvenimento ad esso associato, in modo da avere tutta la cronologia del sistema. Al termine di ogni listening session i messaggi di logging vengono salvati in /logging/logging.txt ed hanno una struttura prefissata in modo da essere facilmente manipolati da un analizzatore di logging apposito.

3 Installazione del tool

WinKnocks fa utilizzo dei seguenti tools:

- La libreria libpcap per Windows, conosciuta come winpcap.
  Nota: winKnocks knocker fa utilizzo di winpcap 3.0
  (perché usa Nemesis packet Injector)
  winKnocks fa utilizzo di winpcap 3.1 (la distribuzione più recente)
- Jpcap - v0.01.16 (la versione più recente): è un set di classi Java che forniscono un'interfaccia per catturare pacchetti dalla rete attraverso winpcap.
- JSyntaxColor 1.2.9: è una libreria per colorare in tempo reale l'input dell'utente su componenti Java-Swing.
- Xerces2 Java Parser 2.8.0: è un parser e schema-processor per files XML.

La distribuzione di winKnocks si presenta come un file zip. La procedura di installazione è composta dai seguenti passi:

1) Estrarre in una directory qualsiasi il contenuto di winKnocks-v1.0.zip;
2) Scaricare e installare le seguenti applicazioni:
   a. Installare winpcap (in accordo con quello che è stato detto sopra) facendo doppio click sul file auto-installante;
   b. Installare Jpcap v0.01.16 facendo doppio click sul file auto-installante;
   c. decomprimere JSyntaxColor zip-file e mettere sc.jar nella directory lib di winKnocks;
   d. decomprimere Xerces zip-file e mettere xercesImpl.jar nella directory lib di winKnocks;

4 Utilizzo del tool

Se sono state eseguite tutte le istruzioni elencate sopra, il tool può essere lanciato eseguendo semplicemente lo script bash "winKnocks.bat". Questa cosa vale sia per la componente Knocker sia per la componente Listener. Il file winKnocks-v1.0.zip contiene due directory: winKnocksClient e winKnocksServer, che corrispondono rispettivamente alle componenti Knocker e Listener del tool. Queste directory sono molto simili, ecco cosa contengono:

- Docs/: contiene questo manuale;
- EclipseProject/: contiene la root directory del progetto Eclipse utilizzato durante lo sviluppo del tool; può essere utile nel caso in cui si voglia modificare al codice sorgente;
- Img/: contiene i files di immagine che utilizza il tool;
- Lib/: contiene le librerie utilizzate dal tool;
- xSchemas/: contiene i files .dtd per validare i files XML delle knock sequences e delle opzioni del tool;
- knockSequences/: contiene i files XML delle knock sequences;
- options.xml: file XML contenente le opzioni del tool; da notare che i files delle opzioni di Knocker e Listener sono totalmente diversi;
- winKnocks[Knocker/Listener].jar: archivio Java che contiene i binari del tool;
- winKnocks.bat: script BASH per avviare facilmente l'esecuzione del tool.

Gli unici elementi per cui differiscono le due directory sono tre. All'interno di winKnocksServer c'è una directory aggiuntiva chiamata "logging" che contiene il file di logging della componente Listener; inoltre c'è anche una directory chiamata "TestBed" che contiene alcuni files per il testing del tool, il suo utilizzo verrà spiegato nel paragrafo 4.2. La terza differenza consiste nella cartella "PacketInjector", presente solo nella directory "winKnocksClient".
La directory "PacketInjector" contiene i files binari del tool Nemesis Packet Injector, che è utilizzato dalla componente Knocker per inviare pacchetti TCP e ICMP. I pacchetti UDP sono inviati attraverso codice Java puro. Si è reso necessario l'utilizzo di un tool esterno per inviare pacchetti TCP e ICMP poiché Java non ha delle librerie adatte a questo scopo, ma soprattutto perché Windows XP con service pack 2 pone delle restrizioni notevoli sull'invio di tali pacchetti. Per chiarire il concetto ecco un estratto del sito degli updates di Windows:

What new functionality is added to this feature in Windows XP Service Pack 2?

Restricted traffic over raw sockets

Detailed description

A very small number of Windows applications make use of raw IP sockets, which provide an industry-standard way for applications to create TCP/IP packets with fewer integrity and security checks by the TCP/IP stack. The Windows implementation of TCP/IP still supports receiving traffic on raw IP sockets. However, the ability to send traffic over raw sockets has been restricted in two ways:

- TCP data cannot be sent over raw sockets.
- UDP datagrams with invalid source addresses cannot be sent over raw sockets. The IP source address for any outgoing UDP datagram must exist on a network interface or the datagram is dropped.

Why is this change important? What threats does it help mitigate?

This change limits the ability of malicious code to create distributed denial-of-service attacks and limits the ability to send spoofed packets, which are TCP/IP packets with a forged source IP address.

Limited number of simultaneous incomplete outbound TCP connection attempts

Detailed description

The TCP/IP stack now limits the number of simultaneous incomplete outbound TCP connection attempts. After the limit has been reached, subsequent connection attempts are put in a queue and will be resolved at a fixed rate. Under normal operation, when applications are connecting to available hosts at valid IP addresses, no connection rate-limiting will occur. When it does occur, a new event, with ID 4226, appears in the system's event log.

Why is this change important? What threats does it help mitigate?

This change helps to limit the speed at which malicious programs, such as viruses and worms, spread to uninfected computers. Malicious programs often attempt to reach uninfected computers by opening simultaneous connections to random IP addresses. Most of these random addresses result in a failed connection, so a burst of such activity on a computer is a signal that it may have been infected by a malicious program.

What works differently?

This change may cause certain security tools, such as port scanners, to run more slowly.

How do I resolve these issues?

Stop the application that is responsible for the failing connection attempts.

In conclusione, la componente Knocker non è in grado (al momento) di inviare pacchetti TCP e ICMP se è utilizzata su sistemi Windows con service pack 2; tutti gli altri sistemi sono pienamente supportati. Comunque l'invio di pacchetti UDP è totalmente supportato, grazie alla portabilità di Java. La componente Listener può essere utilizzata su tutti i sistemi Windows.
Qui di seguito sono descritte nel dettaglio le funzionalità e le interfacce delle componenti Knocker e Listener del tool.

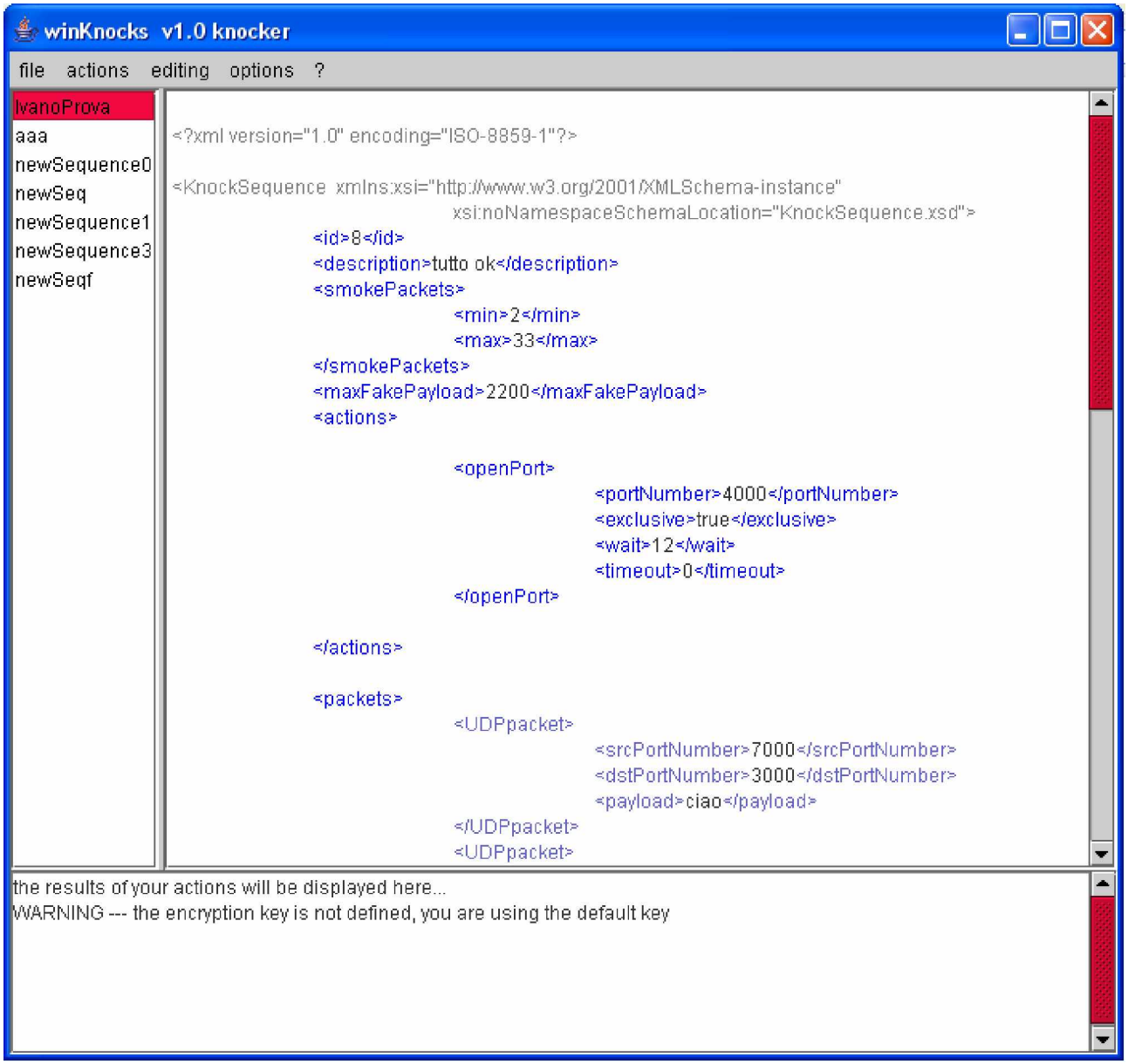
4.1 Funzionalità e GUI del Knocker

Avviando la prima volta la componente Knocker l'utente verrà invitato a settare le opzioni generali del tool, che sono:

- La posizione all'interno del file-system dei binari del tool Nemesis; ovviamente winKnocks controlla automaticamente che tale file esista. Nel caso in cui si vogliano utilizzare solo pacchetti UDP è sufficiente non settare questo parametro; in questo caso verrà generato un *warning* di avvertimento.

- La chiave segreta per criptare i payload dei pacchetti delle knock sequences. Questo parametro può essere una stringa qualsiasi ma, per far funzionare l'intero meccanismo, è necessario che questa chiave sia allineata con quella del Listener.
- L'interfaccia di rete su cui inviare i pacchetti delle knock sequences. WinKnocks rileva dinamicamente le interfacce che sono collegate ad un qualsiasi tipo di rete.

Una volta settate tutte le opzioni, il tool visualizza la sua interfaccia operativa. Quest'ultima è illustrata nella figura sottostante.

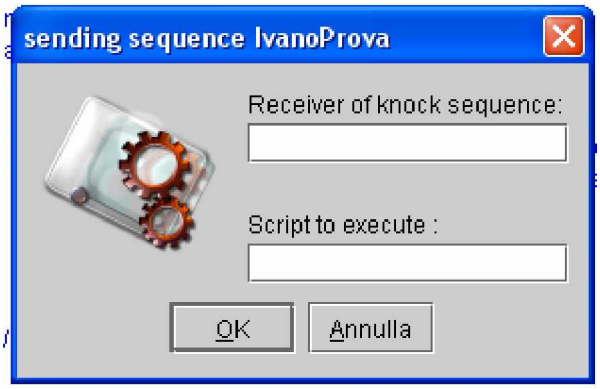


Come si può notare dalla figura, l'interfaccia di winKnocks può essere suddivisa in tre componenti:

- quella più a sinistra contiene l'elenco di tutte le knock sequences definite dall'utente ed è generata dinamicamente all'avvio del tool;
- quella in basso è una console di output e serve per visualizzare tutte le operazioni fatte dall'utente durante l'utilizzo del tool; può essere utile per avere uno schema riassuntivo di tutte le operazioni effettuate;
- La terza componente è un editor XML implementato ad-hoc per creare e modificare i files XML delle knock sequences.

La barra di stato è lo strumento principale del tool ed è costituita da quattro menu:

- *file*: contiene le operazioni tipiche di un editor di testo, che sono: *new*, *edit*, *rename*, *save* e *delete*.
- *Actions*: contiene due sottomenu. Il primo (*send knock sequence*) serve per inviare la knock sequence attualmente selezionata. A questo punto all'utente si presenta il dialog in figura.



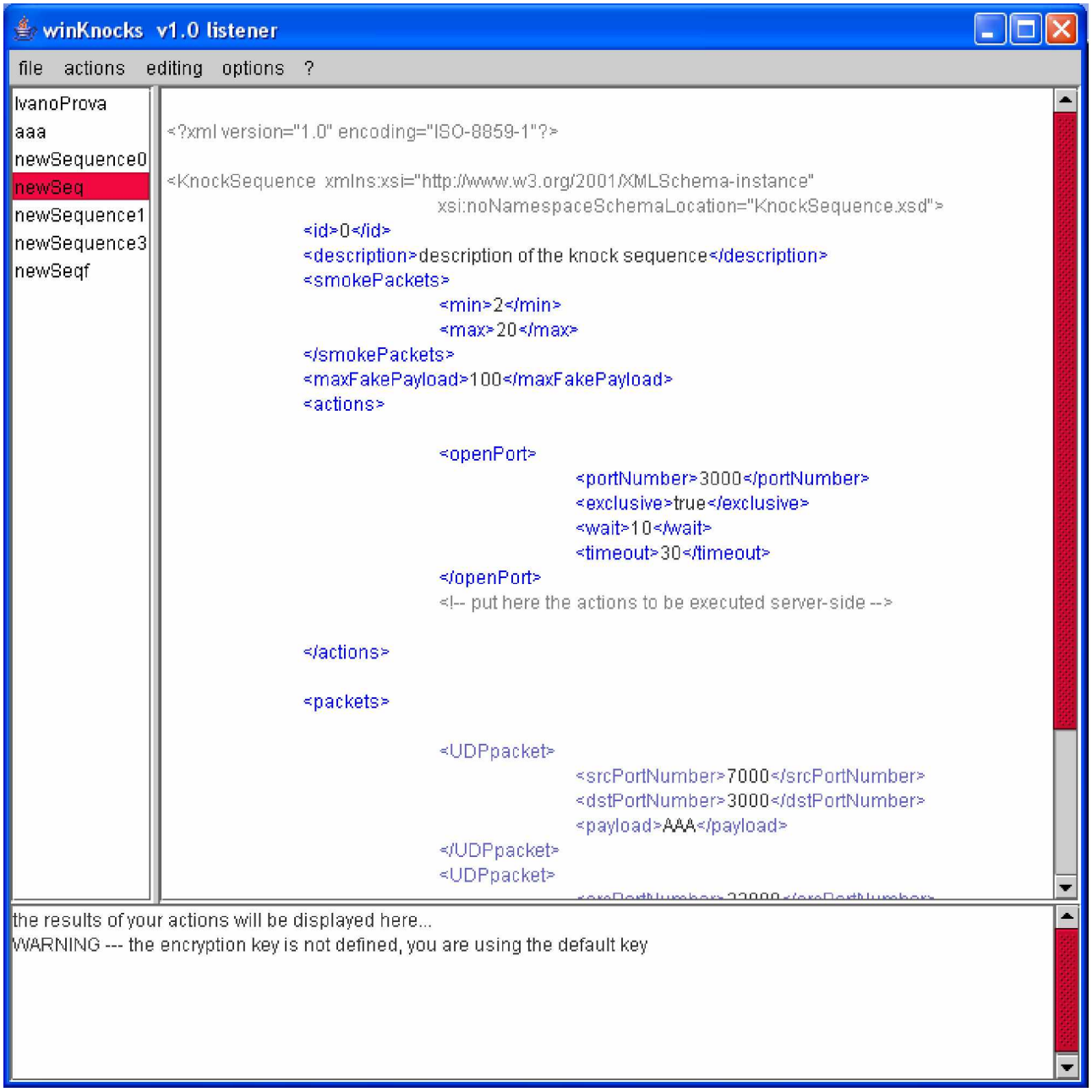
  Nel primo campo l'utente deve inserire l'indirizzo IP o il nome della macchina a cui si vuole inviare la knock sequence. Nel secondo campo (opzionale) si può inserire l'absolute path dello script da eseguire server-side in modalità urgente. Il secondo sottomenu (*refresh list*) permette all'utente di aggiornare la lista delle knock sequences in seguito a qualche cambiamento dei file XML. Questa funzione consente all'utente di usare il suo editor di testo preferito per modificare i files e poi aggiornare il tool con un solo click.
- *editing*: questo menu fornisce delle utilità per creare velocemente delle knock sequences in quanto con un solo click l'utente può aggiungere un intero blocco XML relativo ad azioni e pacchetti.
- *Options*: questo menu fa accedere l'utente alla schermata di modifica delle opzioni del tool; contiene inoltre delle possibili varianti dei colori della GUI di winKnocks.

4.2 Funzionalità e GUI del Listener

Avviando la prima volta la componente Listener l'utente verrà invitato a settare le opzioni generali del tool, che sono:

- Check-box per scegliere se permettere o no l'esecuzione di script in modalità urgente su questa macchina.
- Un campo di testo in cui l'utente può inserire manualmente il filtro della componente che analizza il traffico di rete entrante. Se si desidera utilizzare il filtro automatico è sufficiente "checkare" la check-box appena sopra.
- La chiave segreta per decriptare i payload dei pacchetti delle knock sequences. Questo parametro può essere una stringa qualsiasi ma, per far funzionare l'intero meccanismo, è necessario che questa chiave sia allineata con quella del Knocker.
- L'interfaccia di rete su cui ricevere i pacchetti delle knock sequences.

Una volta settate tutte le opzioni, il tool visualizza la sua interfaccia operativa. Quest'ultima è illustrata nella pagina seguente.



Come si può notare dalla figura, l'interfaccia di winKnocks può essere suddivisa in tre componenti:

- quella più a sinistra contiene l'elenco di tutte le knock sequences definite dall'utente ed è generata dinamicamente all'avvio del tool;
- quella in basso è una console di output e serve per visualizzare tutte le operazioni effettuate dall'utente durante l'utilizzo del tool; può essere utile per avere uno schema riassuntivo di tutte le operazioni effettuate;
- La terza componente è un editor XML implementato ad-hoc per creare e modificare i files XML delle knock sequences.

La barra di stato è lo strumento principale del tool ed è costituita da quattro menu:

- *file*: contiene le operazioni tipiche di un editor di testo, che sono: new, *edit*, *rename*, *save* e *delete*.
- *Actions*: contiene sette sottomenu:
    a. Il primo (*start listening...*) serve per mettere winKnocks in ascolto sull'interfaccia definita nelle opzioni. La componente dell'interfaccia grafica che prima conteneva

l'editor XML adesso mostra in tempo reale i messaggi di logging generati dal sistema.

b. Il secondo sottomenu (*start listening in background...*) esegue le stesse operazioni del primo, solo che l'interfaccia grafica viene chiusa e i messaggi di logging sono stampati nella shell DOS. Questa modalità di "listening" può essere utile nel caso in cui il tool giri su sistemi poco potenti, in cui l'interfaccia grafica potrebbe compromettere fortemente le performance del sistema.
c. *Start listening in test mode* invece è stato creato per testare la componente Listener in modalità off-line. Per utilizzare tale funzionalità infatti l'utente deve fornire un file libpcap (per esempio quelli generati da Ethereal) precedentemente salvato. A questo punto winKnocks simulerà la ricezione dei pacchetti codificati nel suddetto file.
d. *Stop Listening* ferma la ricezione dei pacchetti da parte del tool, genera il file di logging relativo alla sessione appena conclusa e ripristina l'interfaccia grafica alle impostazione di editing.
e. *refresh list* permette all'utente di aggiornare la lista delle knock sequences in seguito a qualche cambiamento dei file XML. Questa funzione consente all'utente di usare il suo editor di testo preferito e poi aggiornare il tool con un solo click.
f. *Check firewall status* mostra (nella console di output) lo stato del firewall di Windows; in pratica serve per vedere se il firewall è attivo oppure no.
g. *Activate Windows firewall* infine serve per attivare/disattivare il firewall di Windows.

- *editing*: questo menu fornisce delle utilità per creare velocemente delle knock sequences in quanto con un solo click l'utente può aggiungere un intero blocco XML relativo ad azioni e pacchetti.
- *Options*: questo menu fa accedere l'utente alla schermata di modifica delle opzioni del tool; contiene inoltre delle possibili varianti dei colori della GUI di winKnocks.

5 Key features e conclusioni

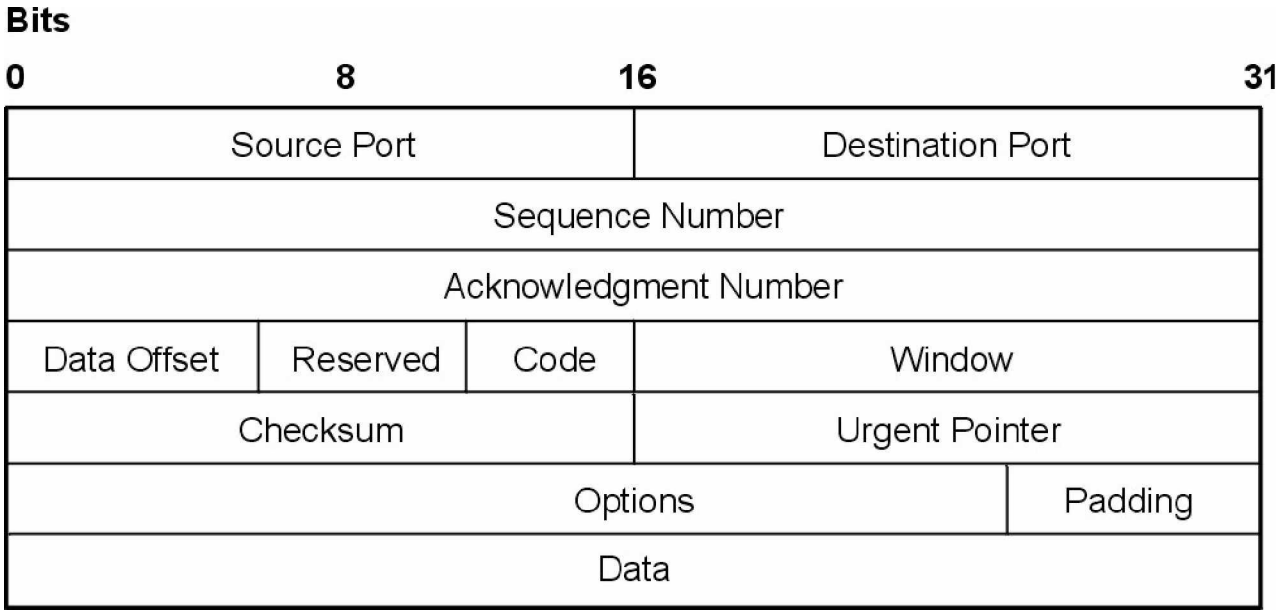
Dopo un'indagine approfondita sui vari tool di portknocking preesistenti, ho cercato di dotare winKnocks di proprietà interessanti.
Per esempio in molti tool le porte del firewall sono aperte o solo all'indirizzo IP che ha fatto la "bussata" o a tutti, ma l'utente non può settare questa caratteristica per ogni singola knock sequence. Nel caso di winKnocks ciò si può fare. Inoltre è l'UNICO tool che si interfaccia direttamente con il firewall di Windows, e che quindi non necessita l'istallazione di programmi aggiuntivi. Flessibilità, cioè in numerosi tools le knock sequences sono di lunghezza prefissata, in winKnocks invece, grazie anche alle capacità di XML, l'utente può definire knock sequences di lunghezza variabile in modo arbitrario. Per esempio potrebbe definire una knock sequences formata da un solo pacchetto per non generare overhead di traffico nella rete, oppure potrebbe generare una knock sequence formata da 100 pacchetti per eseguire le operazioni più sensibili, e cosi via.
Continuando, winKnocks non necessita di un set di porte non utilizzate, ma può stare in ascolto su tutte le porte della macchina: è il filtro che distingue il traffico standard dalle knock sequences.
Inoltre, winKnocks non necessita che tutte le porte del firewall siano chiuse, anzi questo tool può inviare e ricevere knock sequences anche su porte aperte! Questa è una caratteristica molto importante, immaginiamo il seguente scenario: la porta 80 della macchina del Listener ed esso è in ascolto su quella porta; un possibile filtro del Listener potrebbe essere:

```
tcp and tcp-syn == 1 or tcp-fin == 1
```

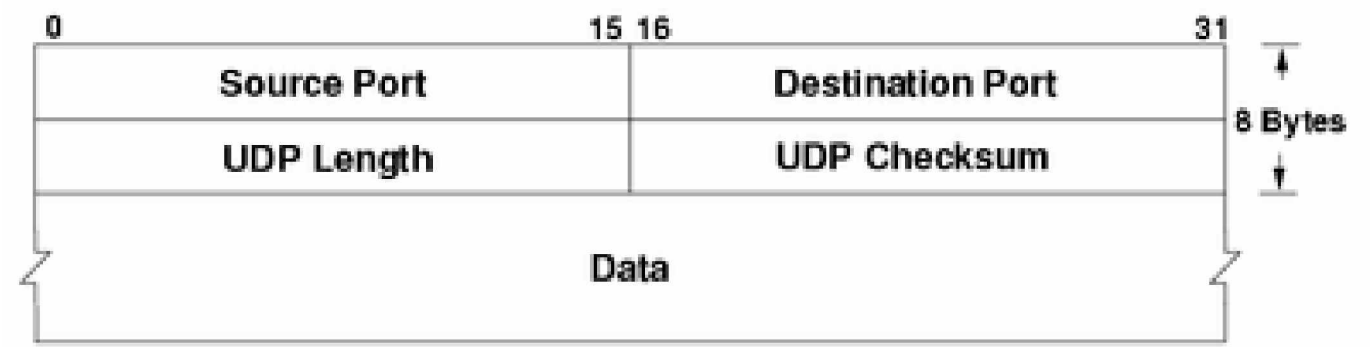
In questo modo sulla porta 80 può esserci traffico qualsiasi, ma il Listener riceve e analizza solo i pacchetti veramente indirizzati a lui. winKnocks analizza i pacchetti in modo trasparente perché essi sono filtrati in modalità non-bloccante; quindi ogni pacchetto, dopo essere stato analizzato, percorre tutto lo stack dell'host che lo ha ricevuto in modo del tutto normale. Un possibile intrusore quindi, cercando di sniffare il traffico di rete, vedrebbe traffico normalissimo sulla porta 80 con richieste e risposte del tutto consistenti. Tutto questo contribuisce allo scopo finale del portknocking: la comunicazione tra Knocker e Listener in modalità totalmente "stealth". Ovviamente winKnocks può essere sottoposto ad attacchi di tipo Denial of Service, ma un filtro creato in modo intelligente potrebbe diminuire fortemente le probabilità di riuscita di tali attacchi. Un possibile miglioramento del tool potrebbe essere quello di creare una componente che identifichi i tentativi di attacchi di tipo brute-force o Denial of Service e reagisca in modo adeguato; in questo modo si diminuisce notevolmente la possibilità che il Listener non sia più in grado di ricevere knock sequences, rendendo impossibile la connessione all'host protetto.

APPENDICE A: header dei pacchetti TCP, UDP, ICMP

TCP



UDP



ICMP

ICMP Header

8	16	32 bits
Type	Code	Checksum
Identifier		Sequence Number
Data		