La tècnica del Port Knocking

Lluís Madrid Claramunt

17 de gener de 2013

Treball Final de Màster

Màster Seguretat de les TIC

Tutor: Sergio Castillo Pérez

Índex

02 Introducció i objectius

Ens trobem en plena era tecnològica en la que l'ús de les tecnologies de la informació ha anat en augment fins a arribar a un punt en que s'ha fet imprescindible per a tothom. La informàtica representa una eina indispensable i cada cop més important per a negocis, governs, estaments, organitzacions i persones. Per tant, podem afirmar que, actualment una proporció elevada d'elements està controlada per equips informàtics (comptes bancaris, satèl·lits, sistemes judicials, comerç, exercits, sistemes sanitaris, ...).

A finals dels anys 80, comencen a aparèixer els primers virus i és realment quan es comença a tenir consciència de la importància de la seguretat informàtica. Segons l'informe anual CSI Computer Crime & Security Survey de 1997, realitzat pel Computer Security Institute1, durant aquell mateix any, el 54% d'empreses dels Estats Units van patir algun atac en els seus sistemes informàtics. Això va provocar, en conseqüència, pèrdues de 137 milions de dòlars.

El fet de que ens puguem trobar tot tipus d'informació en equips informàtics ha fet que els atacs contra aquests hagin augmentat considerablement. Això ha provocat que la seguretat informàtica sigui una eina molt important per a qualsevol sistema informàtic amb informació sensible emmagatzemada. En el cas de la utilització de serveis i aplicacions de xarxa, aquesta necessitat encara és més evident, fins al punt que podem afirmar que la seguretat informàtica és imprescindible.

Dins de la seguretat informàtica existeixen multitud de mètodes, procediments i protocols que intenten que la informació emmagatzemada dins d'un equip informàtic estigui segura. Hi ha cinc característiques fonamentals per a que un sistema sigui segur: Disponibilitat (els sistemes han d'estar accessibles als elements autoritzats), Integritat (els objectes poden ser creats, modificats i esborrats per elements

[1] El **Computer Security Institute** (CSI) és una organització al servei de professionals de la seguretat de la informació.

autoritzats), confidencialitat (la informació serà inintel·ligible per a aquells usuaris no autoritzats), autenticació (només els individus autoritzats tindran accés a la informació) i no repudi (garantir que un usuari negui una operació que ha realitzat).

Segons el model TCP/IP[2], per a poder utilitzar un determinat servei de xarxa, ha d'existir un determinat port, que haurà d'estar obert i a través del qual s'establirà la comunicació. Com ja veurem més endavant, el fet de tenir un port obert pot suposar un forat de seguretat per a qualsevol màquina.

En aquest document, ens centrarem en el Port Knocking, que és una tècnica que es troba dins de l'autenticació. Com veurem més endavant en profunditat, es tracta d'un protocol d'autenticació passiva que, permetrà que un usuari autoritzat, després d'enviar una sèrie de credencials d'autenticació, pugui accedir a un servei que inicialment no estava disponible. Tot i afegir complexitat a la implementació del nostre sistema, ens serà útil per a facilitar la gestió d'aquest i alhora, ens protegirà de possibles atacs. De fet, com ja veurem a l'apartat corresponent, aquesta tècnica ens permetrà accedir a un determinat servei, mitjançant un procés d'autenticació, sense necessitat de que el port d'accés a aquest servei romangui obert. El fet de no tenir necessitat de tenir els ports oberts per a accedir a un determinat servei, en alguns casos no serà útil, però en d'altres, serà fonamental per a preservar la seguretat del nostre sistema i evitar intrusions d'usuaris no autoritzats.

L'objectiu principal d'aquest projecte és el de realitzar un estudi sobre la tècnica del Port Knocking. Per tal d'assolir aquest objectiu, definirem el concepte de Port Knocking i veurem en quines situacions ens convé aplicar aquesta tècnica i quina de les variants s'implementarà depenent de l'escenari en el que ens trobem, solucionant així diversos problemes de seguretat als que podrem fer front.

L'estructura d'aquesta memòria és la següent. En la primera part, tractarem d'introduir el concepte de Port Knocking, descrivint una situació real en la que tenim una necessitat i com, utilitzant la tècnica, podem solucionar la problemàtica. Un cop

[2] El model **TCP/IP** descriu un conjunt de guies de disseny i implementació de protocols de xarxa per a que hi pugui haver comunicació en una xarxa. [2]

introduït el concepte, explicarem en profunditat la seva definició, així com els avantatges i inconvenients d'utilitzar aquesta tècnica. Més endavant, explicarem el seu funcionament bàsic, detallant quin és cada pas i explicant quin és el paper que juga cadascun dels actors durant el procés. A continuació, veurem diferents variants dins del Port Knocking, així com les limitacions de cadascuna i els motius per a utilitzar una o una altra. En el següent apartat, compararem la tècnica amb un sistema similar, com és el SPA, indicant els beneficis i les limitacions de cadascuna. Finalment, veurem implementacions d'eines existents, que duen a la pràctica tot el que hem estat explicant en els apartats anteriors i es realitzarà una exemple pràctic amb una d'aquestes eines existents on veurem detalladament el funcionament del protocol i el paper de cada element.

Per acabar, l'últim punt explicarà les conclusions de tota la informació recollida un cop acabat el treball, així com una breu opinió personal sobre la tècnica.

03 Conceptes bàsics

Començarem aquest capítol introduint alguns conceptes amb els que treballarem durant aquesta memòria per a analitzar en profunditat la tècnica del Port Knocking. Seguidament, veurem com funciona aquesta tècnica, mitjançant un exemple d'ús, situant-nos en un escenari en el que la tècnica del Port Knocking ens pot ser útil. Un cop veiem en quines situacions ens pot ser útil utilitzar la tècnica, veurem la definició de Port Knocking, on explicarem en profunditat en què consisteix la tècnica així com quins són els actors principals i quin és el paper que hi juguen.

A continuació, detallarem quines són les propietats que ha d'acomplir tot sistema per a poder desenvolupar la tècnica del Port Knocking.

Per acabar, veurem quins són els avantatges d'utilitzar el Port Knocking en comparació amb altres sistemes d'accés.

3.1 Conceptes previs

En aquest punt, introduirem una sèrie de conceptes bàsics que ens seran molt d'utilitat al llarg de la memòria. Conceptes com canal de comunicacions, Criptografia, tipus de xifratge, funció resum, tallafocs i la definició formal de la tècnica, així com les seves propietats.

3.1.1 Canal de comunicacions

Un canal de comunicacions és qualsevol medi, ja sigui físic o lògic, que permeti que dos interlocutors o més puguin intercanviar-se informació. Des del punt de vista que ens

interessa, el de la seguretat, podem classificar un canal de comunicacions de la següent manera:

3.1.1.1 Canal de comunicacions segur

Un canal de comunicacions és segur, sempre i quan compleixi les propietats necessàries en tot sistema de xifratge:

- **Autenticació:** El receptor del missatge ha de poder verificar quin ha estat l'origen d'aquest.
- **Confidencialitat:** El missatge, tot i ser interceptat, només podrà ser interpretat correctament pel receptor apropiat.
- **Integritat:** El receptor del missatge podrà verificar que aquest no ha estat modificat des del seu origen.
- **No repudi:** Permetrà provar la participació dels actors dins d'una comunicació. Podem distingir dos tipus:
  - **No repudi a l'origen:** Prova de qui ha estat l'emissor del missatge. D'aquesta manera, l'emissor no podrà negar un missatge que ha enviat.
  - **No repudi al destí:** Prova la recepció del missatge. Així doncs, el receptor d'un missatge no podrà negar haver-lo rebut.

Tot i que tots dos tipus són importants, en un canal de comunicacions segur ens interessarà el primer (no repudi a l'origen) per a poder garantir l'origen dels missatges.

- **Disponibilitat:** El canal de comunicacions ha de permetre la comunicació entre nodes sempre que aquests ho requereixin.

3.1.1.2 Canal de comunicacions insegur

Al contrari que l'anterior, és aquell canal on alguna de les propietats d'un sistema de xifratge no es pot complir. Per a que la comunicació sigui segura en aquests casos, haurem d'utilitzar tècniques se seguretat addicionals, com per exemple, tècniques de criptografia. En el següent punt, veurem com funciona aquesta tècnica.

3.1.1.3 Canal encobert (covert channel)

És aquell canal que permet l'ús de protocols o altres tecnologies que no han estat contemplades en el disseny original, de manera que una transmissió entre dos interlocutors pugui ser indemostrable per a un usuari que estigui capacitat per a analitzar el medi utilitzat.

En aquest tipus de canals, qualsevol camp del paquet que pugui contenir valors aleatoris pot ser utilitzat per a l'establiment d'un canal encobert [8].

3.1.2 Criptografia

Del grec κρυπτός, kryptos, "ocult"; i γράφειν, gráphin, "escriptura". Literalment, vol dir "escriptura oculta". Tot i que el concepte de criptografia té origen al món egipci i la cultura mesopotàmica, està molt present a l'actualitat. Actualment, empreses privades, agències d'intel·ligència, governs, exèrcits, etc. l'utilitzen per a protegir la seva informació [9].

S'ocupa de les tècniques de xifratge que ens ajuden a fer que certs missatges siguin inintel·ligibles per a usuaris no autoritzats que puguin interceptar-los. A més a més, ens permet assegurar l'autenticitat dels missatges i la precisió de les dades rebudes. Per a això, com veurem més endavant, s'utilitzen codis i funcions de firma sobre els missatges, que ens garantiran que aquests no han estat modificats i ens permetran garantir-ne l'origen dels mateixos.

3.1.2.1 Sistema de xifratge

Tot sistema de xifratge ha d'estar composat per:

- **Conjunt finit d' "A"** símbols que permeten formar una cadena de text en clar.
- **Espai de missatges en clar "M".** Serà un conjunt finit de símbols de l'alfabet d' "A".
- **Espai de claus K**, que ens serviran per al xifratge del missatge.
- **Conjunt finit d' "A'"** símbols que permeten formar una cadena de text xifrat.
- **Espai de missatges xifrat "C"**. Serà un conjunt finit de símbols de l'alfabet d' "A'".
- **Espai de claus K'**, que utilitzarem per a desxifrar el missatge xifrat.

I a més a més, ha de permetre realitzar les següents operacions:

- Dues operacions per a realitzar el procés de **xifratge i desxifratge** dels missatges.
- Realització de la **funció resum i firma digital**, per a garantir les propietats de d'autenticació, integritat i no repudi.

3.1.2.2 Tipus de sistemes de xifratge

Podem classificar els sistemes de xifratge en dos grups:

Xifratge simètric o de clau secreta

Tant emissor com receptor coneixeran el valor de la clau o les claus utilitzades per a xifrar/desxifrar els missatges. La clau utilitzada per a desxifrar el missatge (K') pot obtenir-se, en termes computacionals, a partir de la clau de xifratge (K), i a l'inrevés.

Fins i tot, en la majoria dels sistemes de xifratge simètric, la clau de desxifratge és la mateixa que la de xifratge.

Per a que dos interlocutors puguin "entendre's" han d'haver acordat prèviament la clau o les claus a utilitzar. Per a aquest pas d'intercanvi de claus es farà imprescindible un canal segur per a evitar que aquesta informació pugui ser compromesa. El fet de que aquestes claus siguin secretes és la base de la seguretat en aquest tipus de sistemes. Si la clau o alguna de les claus resulta compromesa, qualsevol missatge podria ser desxifrat.

Xifratge asimètric o de clau pública

En aquests sistemes, tots els interlocutors disposaran de dues claus, una pública i una altra de privada. Tant emissor com receptor coneixeran la clau pública (K) de l'altre però, en canvi, tindran una clau privada (K') cadascun d'ells, que només coneixeran ells mateixos. L'emissor xifrarà el missatge amb la clau pública (que coneixen tots els interlocutors). El receptor, en canvi, desxifrarà el missatge amb la seva clau privada (que només coneix ell).

Tot i conèixer la clau pública (K) serà, en termes computacionals, impossible esbrinar la clau privada (K'), utilitzada per a desxifrar els missatges. En canvi, si un usuari coneix la clau privada ha de poder obtenir la clau pública.

L'avantatge d'aquests sistemes respecte als anteriors és que no necessitarem un canal segur per a l'intercanvi de les contrasenyes. La clau que deuen compartir serà pública i podrà estar a l'abast de qualsevol usuari. La clau privada, en canvi, només serà coneguda per cadascun d'ells. Per tant, qualsevol canal serà vàlid per a realitzar l'intercanvi de claus públiques.

El fet de que les claus públiques estiguin a l'abast de qualsevol usuari (en algun directori públic, pàgina web,...) fa que ens haguem de plantejar si aquestes són

autèntiques o han pogut estar modificades per algun altre usuari. Per a pal·liar aquest inconvenient i poder garantir que realment una clau pública pertany a un usuari determinat, farem ús de la PKI (Infraestructura de clau pública). D'aquesta manera, podrem garantir que una clau pública pertany a un determinat usuari (per exemple, mitjançant l'ús de certificats).

3.1.3 Funció resum

Una funció resum o funció Hash, utilitzada per a verificar la integritat de les dades, és aquella funció unidireccional que tindrà una entrada de longitud arbitrària i la sortida tindrà una longitud fixa d' n bits. D'aquesta manera, el resultat d'aquesta funció serà una representació compacta de la cadena d'entrada. Així doncs, podrem trobar-nos amb una mateixa sortida per a dues o més entrades diferents i, en canvi, una mateixa entrada mai tindrà més d'una sortida diferent. [14]

3.1.3.1 Propietats i seguretat d'una funció resum

Per a que una funció hash es pugui considerar segura, ha d'acomplir, com a mínim, les següents propietats:

- **Unidireccionalitat:** Donada una sortida de la funció resum serà impossible, computacionalment parlant, trobar-ne l'entrada.
- **Indistingibilitat:** El coneixement de la sortida no ha d'aportar informació sobre la cadena d'entrada més enllà de la certesa d'acomplir-se la funció. Així doncs, la sortida de la funció resum tindrà una longitud fixa.
- **Facilitat de càlcul**: Ha de ser fàcil poder calcular la funció resum a partir del missatge.

- **Difusió:** La funció resum ha de ser una funció complexa de tots els bits del missatge. Això vol dir que modificant un sol bit de l'entrada (missatge), la funció resum hauria de canviar la meitat dels bits de la sortida, aproximadament.

3.1.4. Tallafocs

Un tallafocs o *firewall* és un element hardware o software que separa en segments la xarxa i permet analitzar el tràfic provinent de cada un d'ells. D'aquesta manera, té la capacitat d'autoritzar o denegar el tràfic depenent de l'origen, destí, protocol,... dels paquets, mitjançant polítiques de seguretat[3] configurades.

3.4.4.1 Propietats d'un sistema tallafocs

Un sistema tallafocs ha d'acomplir, com a mínim, les següents propietats:

1. Tot el tràfic amb origen/destí algun dels segments de la xarxa on opera, ha de passar pel tallafocs. D'aquesta manera, aquest podrà analitzar el tràfic i aplicar les polítiques configurades.

2. Només el tràfic permès a les polítiques del tallafocs podrà travessar-lo.

3. És desitjable que un tallafocs sigui difícilment compromès.

3.2 Exemple d'ús

Tenir una màquina connectada a Internet amb ports oberts cap a l'exterior pot suposar un perill per a la seguretat d'aquesta. Malauradament, hi ha casos en els que els ports

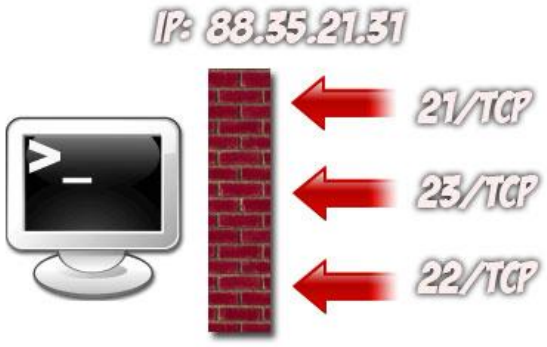
[3] La política de seguretat d'un tallafocs és un conjunt de regles que defineixen quins tipus de paquets poden travessar el sistema i quins no.

hauran d'estar oberts permanentment, com pot ser el cas d'un servidor web, on el port corresponent al servei HTTP haurà d'estar contínuament accessible des de l'exterior.

La majoria de serveis proporcionen autenticació a nivell d'aplicació per a que només usuaris autoritzats que, introduint les credencials correctes, puguin accedir-hi i evitar així crear una porta d'entrada al nostre sistema per a usuaris no desitjats. Hem de tenir en compte que els mètodes utilitzats per a endevinar la contrasenya, sense conèixer-la, normalment no són eficients contra sistemes correctament mantinguts. Tot i això, existeixen multitud d'atacs contra un port obert per a aprofitar vulnerabilitats sense conèixer les credencials necessàries (explotació d'un bug conegut, desbordament de búffer,...). A més a més, hem de ser conscients de que un exploit de día-zero[4] pot aparèixer en qualsevol moment.

En la majoria dels casos, el port corresponent a un servei determinat només haurà d'estar obert en el moment en el que un determinat usuari/servei el necessiti. En aquest tipus de situacions, la tècnica del Port Knocking pot jugar un paper fonamental per a preservar la seguretat del nostre sistema. Aquesta tècnica proporciona una solució efectiva a un dels problemes més greus que actualment afecten als equips informàtics: l'explotació remota de vulnerabilitats.

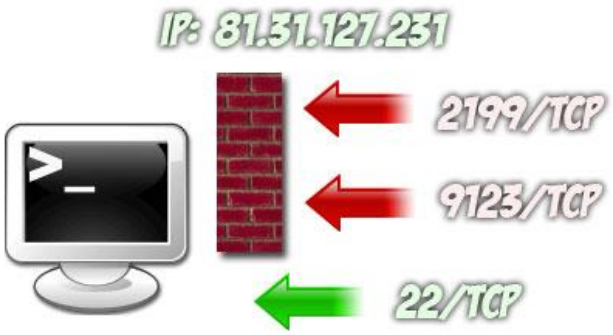
Imaginem un servidor SSH (port TCP/22) tancat, el qual el fa inaccessible des de l'exterior, i per tant protegit enfront exploits i vulnerabilitats. Si un atacant intenta accedir-hi, trobarà que el port està tancat, com a la següent imatge:



[4] Un exploit de día-zero (també és conegut com 0day) és un atac basat en vulnerabilitats que encara no es coneixen i per tant, es desconeix la solució.

Observem que un usuari no autoritzat realitza intents de connexió a determinats ports, i es troba amb que tots ells estan tancats.

Ara bé, si l'usuari autoritzat coneix quina és la seqüència correcta per a realitzar el procés de Port Knocking i per tant, obrir el port, el resultat seria ben diferent:



Observem que l'usuari realitza una seqüència determinada d'intents de connexió que el servidor interpreta. Finalment, el servidor obre el port configurat i l'usuari por accedir-hi i utilitzar el servei en qüestió.

Imaginem un altre cas, en el que necessitem fer ús d'una determinada aplicació, de la qual es coneix una vulnerabilitat evident i que, per algun motiu, no és possible solucionar-la (no té solució, impossible actualitzar el software,...). El fet d'utilitzar la tècnica del PK ens permet fer ús de l'aplicació en qüestió minimitzant el risc de la vulnerabilitat.

3.3 Definició

El **port Knocking** (en català, colpeig de port) és un sistema d'autenticació passiva que, mitjançant un client i després d'enviar les credencials d'autenticació correctes, permet connectar-se a un servei remot a través d'un port que inicialment estava tancat. Com

veurem, aquestes credencials seran intents de connexió a un número determinat de ports en un ordre concret.

Aquesta tècnica, a la que ens referirem com PK al llarg d'aquesta memòria, es basa en el model de comunicacions client-servidor. Com és sabut, tot sistema PK té dos elements diferenciats (un client i un servidor) que s'executen en sistemes diferents. En un model de comunicacions client-servidor, normalment la comunicació entre els elements és bidireccional. Ara bé, en el cas de la tècnica del PK, la comunicació serà unidireccional, sent el primer l'únic que transmet informació. [15]

Un **missatge PK** serà una seqüència vàlida d'intents de connexió que permetrà a un client PK accedir a un servei a través d'un port que està tancat.

Un **client PK** serà l'aplicació capaç de transmetre un missatge PK concret cap al servidor. El seu objectiu serà el d'enviar la seqüència correcta per a que el servidor, després de validar-la, li doni el vist i plau i realitzi les operacions oportunes per a que el client pugui accedir a un determinat servei per un port específic, el qual abans del procés estava tancat.

El **servidor PK** serà l'encarregat d'analitzar els intents de connexió provinents de l'exterior i, en funció de la seqüència rebuda, d'obrir un determinat port. Generalment, s'implementarà com a un procés en segon pla (no precisarà de la intervenció de l'usuari) i monitorarà el tràfic de la xarxa en busca de peticions d'entrada, analitzant cadascuna d'aquestes per a poder identificar la seqüència correcta.

3.4 Propietats

Per a que un sistema de Port Knocking sigui segur, ha d'acomplir una sèrie de propietats [21]:

- **Autenticitat:** El servidor PK serà capaç de verificar que la informació rebuda és autèntica.

- **Confidencialitat:** Tota la informació que intercanvien client i servidor haurà de ser confidencial i, per tant, cap usuari no autoritzat podrà extreure'n informació útil dels missatges intercanviats entre totes dues entitats.

- **Integritat:** El servidor PK ha de poder verificar la integritat dels missatges , és a dir, que els missatges rebuts són els que el client PK ha enviat i, per tant, no han estat modificats.

- **Fiabilitat:** La informació que utilitzarà el servidor PK serà únicament la que el client PK li envia de manera explícita a través del canal que totes dues entitats tinguin establert.

- **Ocultació:** Tota la informació enviada durant el procés d'autenticació entre client i servidor ha de ser indistingible, computacionalment parlant, a paquets equivalents utilitzats per altres objectius.

3.5 Avantatges i beneficis

Una de les tècniques més utilitzades per a mantenir un sistema connectat a una xarxa determinada segur, és el de filtrar els paquets entrants segons la direcció IP d'origen. D'aquesta manera, podem tenir els ports del nostre sistema oberts però en filtrarem els paquets d'entrada per a que només els paquets provinents d'un origen determinat (del qual en confiem) puguin accedir al nostre sistema, denegant tota la resta. Aquest filtratge es pot fer bé des del sistema operatiu o des d'algun sistema extern com pot ser un sistema tallafocs. Ara bé, aquesta tècnica pot resultar inútil si no coneixem l'origen de les possibles peticions o, l'origen pot ser qualsevol. Si, a més a més, hi sumem el fet de que un atacant pot fet IP spoofing[5] o inclús, l'atacant podria accedir a alguna de les màquines amb direccionament permès i, per tant, tindria accés al port del nostre sistema.

[5] L'**IP spoofing** es produeix quan un atacant suplanta la direcció IP d'un altre usuari de la xarxa.

El benefici principal i fonamental de la tècnica de PK és la de mantenir un determinat port, que necessitarem per a accedir a un determinat servei, tancat mentre no haguem d'utilitzar-lo. D'aquesta manera, el port roman obert no més temps del necessari i així evitem que estigui exposat a l'exterior de manera innecessària. Així doncs, amb la utilització de la tècnica del Port Knocking ens estalviem la limitació de filtrar determinades xarxes. A més a més, al contrari que amb la tècnica de filtratge per direcció IP d'origen, on si la màquina permesa era compromesa per un atacant, aquest podria tenir accés al port de la nostra màquina, amb el PK l'atacant, tot i aconseguir comprometre una determinada màquina, haurà de conèixer, condició sine qua non, les credencials per a poder activar les regles del sistema tallafocs. Més endavant veurem que existeixen tècniques d'atac que permetran a un atacant comprometre la comunicació en un sistema que utilitzi la tècnica del Port Knocking i aprofitar-se, sense conèixer, a priori, les credencials.

Un altre avantatge important és que la informació entre client PK i servidor PK (missatge PK) té forma d'intents de connexió i no, de paquets amb carga útil. D'aquesta manera, un atacant que desconegui l'existència d'aquesta tècnica en el sistema, mai podrà accedir-hi. Inclús sabent que s'utilitza la tècnica de PK al sistema, l'atacant necessitaria un gran esforç de força bruta per a descobrir una seqüència mínima de tres intents, havent de provar totes les combinacions possibles[6]. Això equival a 65535^3 (281 bilions) d'intents de connexió en el pitjor dels casos. Uns 140 bilions de paquets en mitjana, per a aconseguir esbrinar una seqüència mínima de tres ports. A més a més, l'atacant hauria d'estar contínuament escanejant tots els ports per a esbrinar quin d'ells és el que s'obre.

Un altre avantatge és que el software necessari per a dur a terme la tècnica PK, tant en client com en servidor, és molt senzill. D'aquesta manera, l'impacte que té la tècnica sobre el consum de la CPU i el tràfic és mínim.

[6] El rang de ports d'un sistema és de 1-65535.

04 Funcionament bàsic

En aquest apartat, explicarem com funciona la tècnica del Port Knocking pas a pas i quin és el paper que hi juguen tots els actors implicats en cada moment.

La tècnica del Port Knocking es pot dividir, bàsicament, en tres passos ben diferenciats. En el primer pas, el servidor PK estarà configurat de tal manera que escoltarà totes les peticions entrants i podrà detectar-ne les vàlides. Seguidament, el client PK iniciarà el procés d'autenticació i, un cop autenticat correctament, el servidor PK serà l'encarregat de dur a terme l'acció que li ha estat configurada, com per exemple, obrir un determinat port per a que una aplicació externa, a través del client PK, pugui accedir-hi.

4.1 Detecció en servidor PK

4.1.1 Què és un servidor PK?

Un servidor PK serà el procés encarregat d'analitzar el tràfic de xarxa entrant al sistema. Es tracta d'un procés independent i que, un cop configurat, no requereix l'actuació de l'usuari més enllà de la supervisió del seu estat, ja que és un procés dimoni[7]. Segons unes polítiques de seguretat configurades al servidor, autoritzarà o no el tràfic.

Un servidor PK pot estar programat en qualsevol llenguatge i pot utilitzar-se per a qualsevol Sistema Operatiu, sempre i quan pugui desenvolupar les següents tasques:

1) Analitzar el tràfic d'entrada al servidor i detectar qualsevol intent de connexió procedent del client PK.

2) Validar els intents de connexió, verificant la seva integritat.

[7] Un **procés dimoni** o **daemon** és aquell que s'executa en segon pla i no requereix la intervenció de l'usuari.

3) Realitzar les accions oportunes que han estat configurades a les polítiques de seguretat del sistema.

4.1.2 Tipus de detecció

Com ja hem comentat, el servidor PK estarà darrere del firewall i, per tant, aquest mantindrà tots els ports filtrats per a que romanguin tancats des de l'exterior del sistema. Però si el firewall bloqueja tots els ports, com podrà el servidor PK detectar els intents de connexió? Necessitarem algun mecanisme que permeti al Servidor PK poder accedir a la informació saltant-se el bloqueig de ports.

Per a poder analitzar el tràfic d'entrada cap al servidor, el servidor PK serà configurat d'una de les dues opcions següents, depenent del criteri de l'administrador de la xarxa.

4.1.2.1 Tràfic de xarxa

Abans d'explicar com funciona aquesta tècnica explicarem què és el mode promiscu i com es comporta una targeta de xarxa quan està en aquest mode, per tal d'entendre com el servidor pot processar paquets tot i que el firewall bloquegi tots els ports.

Quan una targeta està en mode promiscu captura tot el tràfic de la xarxa (vagi dirigit cap a ella o no). Quan un paquet arriba a la targeta, el driver[8] d'aquesta envia una còpia d'aquest a la controladora adequada i, si està en mode promiscu, addicionalment envia una còpia del paquet al packet-filter[9]. Per tant, l'aplicació que està realitzant la captura tindrà accés a una còpia de tots els paquets que circulin per la xarxa, inclosos aquells que seran descartats per direcció IP destí diferent o aquells que seran descartats per no superar les polítiques del firewall. D'aquesta manera, ens assegurem que, tot i que el firewall bloqueja paquets cap a determinats ports, el servidor PK podrà "escoltar" tots els intents de connexió que es realitzen.

[8]Un **driver** o **controlador de dispositiu** permet al Sistema Operatiu interactuar amb un perifèric.
[9]El **packet-filter** s'encarrega d'enviar la còpia del paquet a l'aplicació que ho ha demanat.

Tot i que és una tècnica molt eficient, implica una necessitat addicional de recursos de la màquina per a poder gestionar tots els paquets que circulen per la xarxa.

4.1.2.2 Fitxer log

Una altra alternativa a l'hora d'analitzar els intents de connexió cap al servidor PK és la d'utilitzar el fitxer log[10]del firewall per a registrar totes les peticions que es realitzen cap als ports del servidor PK. Podem configurar el firewall per a que reflecteixi en aquests registres informació tant dels paquets que passen el sistema de polítiques com dels que són denegats. D'aquesta manera, el servidor PK només hauria de consultar el registre log per a saber si algú està realitzant peticions a algun dels seus ports o d'altra manera, configurar el Firewall per a que enviés un missatge al servidor PK cada cop que algú realitza un intent de connexió amb ell.

És una tècnica molt simple de configurar però també ineficient. El temps que pot transcórrer entre que succeeix un esdeveniment fins que aquest es registra al log és indeterminat i no està al abast de l'administrador de la xarxa sinó de l'electrònica i de la configuració del sistema, ja que aquests dos factors incidiran directament en el temps de processament del log.

4.2 Comunicació del client cap al servidor

4.2.1 Missatge del client cap al servidor

En aquest pas, el client PK envia un missatge determinat cap al servidor, per a que aquest en verifiqui la seva autenticitat i realitzi les accions oportunes que prèviament s'han configurat al sistema. Com ja s'ha fet referència a l'apartat *3.4-Propietats*, el sistema ha d'acomplir amb totes les propietats indicades (autenticitat, confidencialitat,

[10]Un fitxer **log** és aquell que reflexa un registre d'events durant un temps determinat.

integritat, fiabilitat, i ocultació). S'hauran d'acomplir totes les condicions anteriors i, a més a més, el canal a través del qual es realitza la comunicació haurà de permetre una comunicació unidireccional (Client → Servidor).

En aquest document només ens centrarem en el desenvolupament i implementació de la tècnica del Port Knocking utilitzant protocols TCP/IP.

Com ja veurem amb més detall a l'apartat *5 – Variants existents*, la informació que envia el client PK cap al servidor PK seran paquets amb els camps de les capçaleres TCP/IP modificats, depenent de quina informació vol enviar i com està configurat el servidor PK (port origen, port destí, seqüència de ports, temps de vida,...).

4.2.2 Tipus de missatge

El missatge que enviarà el client PK cap al servidor PK serà més o menys complex segons el nivell de seguretat que desitgem.

4.2.2.1 Missatge sense autenticació

De tots els casos possibles, aquest és el més senzill. En aquest escenari, l'autenticació es basa en el "secret" de la seqüència de ports destí dins dels paquets que envia el client PK. D'aquesta manera, un client PK enviarà diversos paquets amb diferents ports destí i, el servidor PK l'autenticarà quan observi que aquests ports destí coincideixen amb els que aquest té configurats.

Des d'aquest punt de vista, el servidor PK haurà d'autoritzar l'accés a tots els ports per tal de poder conèixer el número de port al qual el client desitja connectar-se. El servidor PK autoritzarà la IP origen d'aquestes peticions assumint que és qui desitja obrir un port determinat per a connectar-se a un determinat servei.

L'inconvenient principal d'aquesta tècnica és que tot el tràfic viatja en clar. Per tant, un atacant que estigui escoltant el tràfic entre client-servidor serà capaç d'esbrinar el

secret molt fàcilment. Així doncs, un cop l'atacant coneix el secret, podrà crear tantes autenticacions com desitgi, generant els mateixos paquets i modificant la IP origen. D'altra banda, el fet de no poder fixar un port determinat ens obliga a haver d'autoritzar 2^{16} ports.

La trivialitat de la implementació d'aquest sistema fa que s'utilitzi bastant sovint, tot i la seva escassa seguretat. Alguns exemples d'implementació amb aquest sistema, són:

Cd00r, packet coded backdoor [22]

Knockd - a port-knocking server [23]

La escassa seguretat en aquest tipus d'implementacions xoca frontalment amb el principi de *Kerckchoffs*[11] [24], punt número 2 descrit a continuació, per a un sistema de xifratge:

- Si el sistema no és teòricament invulnerable, com a mínim ho haurà de ser a la pràctica.
- L'efectivitat del sistema no ha de dependre de que el seu disseny romangui en secret.
- La clau haurà de ser fàcilment memoritzable de manera que no calgui recórrer a notes escrites.
- Els criptogrames han de donar resultats alfanumèrics.
- El sistema ha de ser operable per una única persona.
- El sistema ha de ser fàcil d'utilitzar.

Més endavant, el principi de *Kerckchoffs* va ser reformulat per *Claude Shannon*[12] i es va convertir en un dels principis més utilitzats pels criptòlegs:

[11] **Auguste Kerckshoffs** (1835,1903) va ser un criptògraf holandès i va ser el primer en parlar sobre criptografia militar. Va concretar els sis principis bàsics per al disseny de sistemes criptogràfics.

"L'adversari coneix el disseny del sistema. Per tant, la seguretat d'aquest recaurà en la clau."

El fet de que la clau sigui fàcilment coneguda per qualsevol atacant és una de les principals raons per les quals es diu que la tècnica del Port Knocking simplement només aporta *seguretat via obscuritat*[13] i s'oposa totalment al principi de Shannon.

Com veurem en els següents tipus de missatges, no serà així en totes les implementacions.

4.2.2.2 Missatge simple sense autenticació

Aquesta tècnica intenta ser tan simple com l'anterior però solucionant-ne algun dels inconvenients. Es generarà un missatge simple amb els següents camps:

- Direcció IP que es desitja autoritzar.
- Port al qual es desitja accedir.

A més a més, addicionalment hi podrem afegir un camp addicional (acció a realitzar) que ens donarà més marge d'acció:

- Acció a realitzar (obrir o tancar el port).

Tot i que amb aquest missatge el servidor ja disposa de tota la informació suficient, la informació no proporciona cap tipus d'autenticació. El client pot enviar aquest missatge en clar o xifrat. En aquest últim cas, tot i impedir que un atacant pugui accedir a la informació que es transmet a la xarxa, no realitza cap tipus d'autenticació o integritat. Per tant, si un cop desxifrat el missatge, el servidor no és capaç d'interpretar-lo, no podrà determinar si aquest és autèntic o no.

[12] **Claude Elwood Shannon** (1916,2001) va ser un enginyer electrònic i matemàtic. Conegut com el "pare de la teoria de la informació".

[13] El concepte de **seguretat via obscuritat** és un controvertit principi de seguretat que utilitza el secret per a garantir la seguretat. Es considera un procés vàlid però insuficient per a la seguretat d'un sistema.

Aquest tipus de missatge, com en el cas anterior, també xoca frontalment amb els principis de *Kerckchoffs* i *Shannon*, ja que no consta de cap tipus de seguretat i tota la informació viatja en clar.

4.2.2.3 Missatge amb Integritat

Una possible solució al problema de l'anterior tècnica, el de no poder verificar la integritat del missatge, podria ser el d'afegir al missatge un camp que permeti al servidor PK distingir-ho d'un conjunt de bits aleatoris. Això és possible modificant determinats camps de les capçaleres dels paquets TCP/IP. Aquest camp serà una *suma de comprovació*[14] o un *hash criptogràfic*[15]calculat a partir de tota la informació del missatge. D'aquesta manera, el servidor serà capaç de distingir missatges vàlids de missatges que continguin informació aleatòria.

Tot i aquest element afegit, que permetrà al servidor verificar la integritat del missatge, aquest podrà ser enviat, com en el cas anterior, en clar o xifrat. En aquest segon cas, el fet de que el missatge estigui xifrat i, a més a més, contingui el camp de comprovació d'integritat del missatge fa que el fet de modificar el missatge sigui molt més complicat per a un atacant. A més a més, acompliria els principis de Kerckchoffs i Shannon i, per tant, passaria de ser una tècnica de seguretat via obscuritat a ser una tècnica de seguretat robusta.

4.2.2.4 Missatge amb Integritat i Autenticació

Finalment, comentarem l'opció més completa de totes. A la tècnica anterior, se li pot afegir un camp que permeti verificar la integritat del missatge i, alhora, permeti al client PK autenticar-se amb el servidor PK. L'esquema és molt semblant a l'anterior

[14]La **suma de comprovació** és un mètode senzill de detecció i correcció d'errors molt eficient amb cadenes de dades de longitud petita.
[15]El **hash criptogràfic** és una funció unidireccional que relaciona una determinada entrada amb un valor dins d'un conjunt finit.

amb la variant que, enlloc d'utilitzar una suma de comprovació o hash, utilitzarem un codi d'autenticació de missatge (MAC)[16].

Per a calcular la funció MAC del missatge es realitza una funció resum criptogràfica amb una determinada clau secreta que només coneixeran client i servidor. Aquesta funció resum ha de ser criptogràfica per la senzilla raó que ha d'acomplir certes propietats de seguretat per a poder resistir possibles atacs. D'aquesta manera, la funció MAC tindrà dos arguments d'entrada: una determinada clau i un missatge de longitud indeterminada. El resultat de la funció serà un codi MAC de longitud fixa.

D'aquesta manera, el missatge podrà ser enviat en clar pel client PK garantint que cap atacant el pugui modificar.

4.3 Gestió del servidor

Un cop el servidor comprova que efectivament, ha rebut un missatge correcte per part d'un client PK vàlid, executarà una acció determinada que prèviament haurem d'haver configurat a les polítiques de seguretat del firewall del servidor. Quan es parla de Port Knocking, generalment es fa referència a l'objectiu d'obrir un determinat port que prèviament estava tancat però tanmateix, hi ha una varietat infinita de possibilitats que es pot configurar dins del servidor com a resposta d'aquest a un missatge vàlid:

- Obrir un port/conjunt de ports.
- Tancar un port/conjunt de ports.
- Execució d'un script.
- ...

[16] **MAC** (Message Authentication Code) és una porció d'informació utilitzada per a autenticar un missatge [25].

4.4 Finalització de la comunicació

Després que el client PK hagi enviat el missatge corresponent al servidor PK, i aquest últim l'hagi validat correctament i per exemple, obert el port, una aplicació externa, a través del client PK, ja tindrà accés al servei sol·licitat. Un cop finalitzada la feina, el servidor PK es por configurar de dues maneres diferents per a que aquest port no romangui obert. També podria no configurar-se el tancament del port, però estaríem deixant un forat de seguretat al nostre sistema, fet pel qual estem configurant la tècnica del Port Knocking.

4.4.1 Tancament per time-out

La primera tècnica, i més habitual, seria la de configurar un time-out de tal manera que quan el servidor no detecti activitat en x minuts sobre el port recentment obert es tanqui. És la manera més eficient d'evitar possibles oblits. D'altra manera, un atacant que estigués monitorant la xarxa podria aprofitar-se d'un descuit d'un usuari, que després d'obrir un port determinat mitjançant la tècnica del Port Knocking, se n'oblida i el deixa obert. Tota la seguretat que fins ara havíem configurat al nostre sistema se n'aniria en orris.

Tot i així, un atacant que estigués capturant el tràfic d'un usuari en un determinat port, podria aprofitar el temps entre que l'usuari deixa d'utilitzar-lo i salta el límit del time-out. D'aquesta manera, l'atacant tindria x minuts per a generar tràfic sobre el port obert i aquest romandria sempre obert.

Un altre possible cas en el que aquesta tècnica no ens seria útil és en aquelles situacions en les que obrim un port que és utilitzat per una aplicació que està contínuament generant tràfic (actualitzant o sincronitzant una gran quantitat de dades, realitzant certes peticions a l'exterior cada x minuts, ...) o que és sensible a rebre peticions contínuament. En aquestes situacions, tampoc funcionaria aquesta

tècnica ja que el sistema, en detectar activitat contínua al port, no activaria el time-out.

Tot i ser una tècnica senzilla i molt útil, hem vist que hi ha situacions en les que no la podrem utilitzar. En aquests casos, podrem configurar la tècnica de *tancament del port per PK.*

4.4.2 Tancament per PK

Com hem vist a l'anterior apartat, hi ha situacions en les que la tècnica de tancament del port per time-out no ens serà útil. Així doncs, podrem utilitzar una altra tècnica, no tan senzilla de configurar, però que serà més segura i ens assegurarà el correcte tancament del port que hem estat utilitzant.

Aplicant el mateix procediment que s'ha utilitzat per a obrir un determinat port, el client PK enviarà un missatge al servidor PK per a que aquest tanqui el port. Això es pot fer amb una altra seqüència de ports o configurant el servidor PK per a que una mateixa seqüència de ports es comporti d'una manera quan el port està obert i d'una altra quan està tancat.

D'aquesta manera, l'usuari utilitzarà la tècnica del PK per a obrir el port. Un cop obert el port, s'hi connectarà i farà ús del servei corresponent. Un cop acabada l'activitat, utilitzarà de nou la tècnica del PK i tancarà el port en qüestió per a evitar possibles atacs.

05 Variants existents

Com s'ha comentat en apartats anteriors, tant client PK com servidor PK poden estar programats en qualsevol llenguatge i funcionant en qualsevol Sistema Operatiu, sempre i quan acompleixin una sèrie de requisits que permetin desenvolupar les funcions de cadascun dels elements. Això dóna a l'administrador una llibertat pràcticament absoluta a l'hora d'implementar les eines necessàries per a dur a terme la tècnica del Port Knocking.

En aquest apartat, explicarem algunes de les opcions i variants disponibles que hi haurà a l'hora d'implementar la tècnica segons les circumstàncies i el criteri de l'administrador de la xarxa.

Com veurem a continuació, la tècnica del Port Knocking pot funcionar sobre diferents protocols. Segons la complexitat i els requeriments que es necessiti configurar, un protocol serà més adient que un altre. Només en comentarem algunes de les possibles implementacions.

5.1 ICMP

De tots els protocols que veurem, aquesta és la implementació més senzilla de totes [41], ja que el seu funcionament és molt bàsic.

El client PK enviarà un paquet que utilitzi el protocol ICMP, com per exemple un ping, al servidor PK, que en rebre'l, executarà la regla de seguretat configurada a les seves polítiques de seguretat (per exemple, obrir un port determinat). Aquesta serà una regla molt senzilla que, en rebre un paquet ICMP de la direcció IP origen del client PK, executarà la regla configurada.

Aquesta tècnica està molt limitada ja que l'ICMP és un protocol molt senzill i sense gaires paràmetres a configurar. Tot i així, podem configurar determinats camps del protocol ICMP per a poder distingir un ping normal d'un utilitzat per a la tècnica del PK. Per exemple, podem configurar per a que els pings utilitzats a la tècnica del PK hagin de ser d'un tamany determinat, diferent al que s'utilitza per defecte. D'aquesta manera, quan el client PK vulgui obrir un determinat port, enviarà un ping d'un determinat tamany, per exemple 5000 bytes. D'aquesta manera, i tenint en compte que les capçaleres ocupen 28 bytes, quan el servidor PK rebi un paquet ICMP de 5028 bytes procedent del client PK, executarà la regla configurada a les polítiques de seguretat.

Es tracta de la implementació més senzilla de totes i a l'hora la que menys seguretat aporta al nostre sistema, sense cap tipus de verificació d'identitat ni d'autenticació. Qualsevol atacant que estigués analitzant el tràfic de la xarxa podria veure que, amb un paquet ICMP d'un tamany determinat procedent del client PK, es pot obrir un port del servidor. A més a més, seria molt fàcil per a un atacant, com ja veurem en el següent apartat, realitzar un atac per repetició (capturar el paquet ICMP del client PK i transmetre'l quan vulgui) i aconseguir que s'obrís el port en qüestió.

5.2 UDP/TCP

La utilització dels protocols TCP/IP, tant UDP com TCP, ens proporcionarà un ventall immens de possibilitats respecte a la tècnica anterior, tot i que evidentment, augmentarà considerablement la complexitat de la implementació del sistema.

En els protocols TCP/IP [36,37], qualsevol camp de les capçaleres que pugui tenir valors aleatoris podrà ser utilitzat per a codificar el missatge PK. Així mateix, depenent de la situació en la que ens trobem, serà més adient realitzar una implementació amb UDP o amb TCP.

UDP[17] és un protocol més senzill i que no està orientat a connexió, això implicarà que la implementació serà més senzilla. A més a més, el tràfic UDP pot passar més desapercebut per a un atacant. Per exemple, realitzar un traceroute UDP a un port determinat del servidor PK.

TCP[18], en canvi, és un protocol orientat a connexió que té mecanismes de fiabilitat per assegurar que la comunicació es dugui a terme i els paquets arribin al servidor PK en el mateix ordre en el que han estat enviats. El fet d'utilitzar aquest protocol ens permet afegir mecanismes de control de transmissió dels missatges (control d'errors, mecanismes de seguretat, control de flux,...) però en canvi, afegeix més complexitat a la implementació.

Com hem comentat anteriorment, tant TCP com UDP ens permeten modificar camps de les capçaleres. Aquests camps seran utilitzats pel client PK i pel servidor PK per a intercanviar-se el missatge PK intentant fer-ho el més transparent possible de cara a possibles atacants que estiguin analitzant el tràfic de la xarxa. A continuació, veurem alguns dels exemples de la utilització de les diferents capçaleres dels paquets TCP/IP.

5.2.1 Camp port destí [TCP/UDP]

El camp *port destí* (target port), de longitud 16 bits, pertany a la capçalera de TCP (RFC 793) i UDP (RFC 768), i en aquest es defineix el número del port destí al qual l'emissor desitja enviar un paquet. Per tant, el client PK pot codificar un missatge M i afegir-lo a aquest camp i enviar-li al servidor PK, qui reconstruirà el missatge a partir del camp *port destí.* D'aquesta manera client PK i servidor PK utilitzen la seqüència de ports (dins del camp *port destí*) com a vehicle del missatge PK.

El principal avantatge d'utilitzar aquesta tècnica és la facilitat que té l'usuari per a poder modificar el camp *port destí*, ja que tenim una gran varietat de programes que

[17] **UDP** (User Datagram Protocol) protocol de transport no orientat a connexió.
[18] **TCP** (Transmission Control Protocol) protocol de transport orientat a connexió.

ens permeten escollir a quin port enviar cert tràfic (telnet, hping, navegadors web, clients FTP, ...).

Un altre avantatge a tenir en compte en aquesta tècnica és que aquest camp no es veu modificat per elements de la xarxa (routers, switchs, firewalls, ...) ni per passarel·les a nivell d'aplicació o similar.

Tot i així, també té certs inconvenients. El fet d'utilitzar programes "externs" per a generar tràfic ens obliga a anar amb compte amb els reintents de connexió que aquests realitzen. Podria ser que el programa que estem utilitzant, en detectar que el primer intent de connexió ha fallat, reintentés establir la connexió al mateix port. D'aquesta manera, la seqüència que el client PK envia es veuria alterada i per tant el servidor PK desestimaria l'intent de connexió per haver rebut una seqüència no vàlida.

El fet de que, utilitzant el protocol UDP, els paquets que envia el client PK puguin desordenar-se pel camí, suposa un altre inconvenient d'aquesta tècnica, ja que la seqüència enviada no serà la mateixa a la seqüència que el servidor PK rep.

Un altre inconvenient que hem de tenir en compte a l'hora d'utilitzar aquesta tècnica és el d'haver de travessar dispositius com firewalls intermedis que puguin estar abans d'arribar al servidor. En aquests casos, haurem de permetre el tràfic als ports que corresponguin a la seqüència.

5.1.2 Camp port origen [TCP/UDP]

Aquesta tècnica és molt similar a l'anterior. Es tracta d'un camp de les capçaleres TCP i UDP amb una longitud de 16 bits en el qual s'inclou el número de port des del qual la màquina origen envia el paquet.

El fet d'utilitzar aquest camp enlloc de l'anterior suposa un avantatge a l'hora d'haver de travessar firewalls. En la majoria dels casos, els firewalls bloquegen paquets segons el port destí del paquet procedent de l'exterior enlloc del camp port origen. Així doncs,

aquest camp serà transparent per als dispositius intermedis que ens puguem trobar pel camí.

Tot i així, a diferència de l'anterior tècnica, l'ús d'aquesta té l'inconvenient de requerir eines especialitzades o inclús permisos d'administrador per a poder alterar el valor del camp port origen. A més a més, la utilització de protocols com el NAT en dispositius intermedis pot alterar el valor del camp del port origen, modificant així el valor del missatge que estem enviant.

5.1.3 Camp número de seqüència [TCP]

Com ja hem comentat, TCP és un protocol orientat a connexió que garanteix que les dades son enviades d'un extrem de la comunicació a l'altre sense errors i en el mateix ordre en el que s'han enviat. Per aconseguir aquest últim punt, utilitza l'anomenada finestra TCP, que és un búffer que s'encarrega de que el destí rebi la seqüència de paquets en l'ordre correcte i també de detectar duplicitat en la transmissió dels paquets. El protocol utilitza el camp número de seqüència per a poder portar un control de tots els paquets que s'envien durant la comunicació.

Tot i que en un inici el camp número de seqüència anava augmentant seqüencialment, es va variar aquest comportament en detectar un greu problema de seguretat que permetia el segrest de les sessions TCP [40]. Així doncs, per a evitar aquest problema de seguretat, actualment s'utilitzen números de seqüència pseudo-aleatoris. Aquest fet permet la utilització del camp ISN[19] de 32 bits, que el client PK podrà utilitzar per a enviar-li el missatge al Servidor PK.

L'avantatge d'utilitzar aquesta tècnica respecte totes les anteriors és que aquest camp és transparent per a tots els elements intermedis que ens podem trobar i, per tant, no es veurà afectat.

[19] **ISN** (Initial Sequence Number) és el primer número de la seqüència.

Un altre avantatge respecte a les tècniques comentades anteriorment és que el client PK pot utilitzar els 32 bits del camp per a codificar el missatge cap al servidor PK. Això és el doble que en els casos anteriors.

Un inconvenient a l'hora d'utilitzar aquesta tècnica és que el client PK necessitarà utilitzar eines avançades per a poder modificar aquest camp de la capçalera TCP/IP.

En el cas d'existir servidors *proxies transparents*[20] entre client i servidor, fa que el camp número de seqüència es vegi alterat, ja que el client PK iniciarà una sessió TCP amb el servidor *proxy*, que a l'hora iniciarà una altra sessió TCP independent de la primera amb el servidor PK. Per tant, el missatge PK que el client PK envia cap al servidor PK es veuria totalment modificat.

Com ja hem vist, hi ha moltes maneres d'implementar el sistema per al desenvolupament de la tècnica del Port Knocking i totes són vàlides depenent de les circumstàncies en les que ens trobem. Per tant, els requeriments del sistema (seguretat, complexitat de la implementació,...) i el criteri de l'administrador seran els motius d'escollir-ne una variant o una altra.

[20] Un **proxy transparent** s'utilitza per a que totes les connexions de les màquines que hi ha darrere siguin enrutades a través d'ell. S'utilitza per a seguretat, rendiment, anonimat, etc.

06 Comparació amb SPA (Single Packet Authorization)

En aquest apartat, compararem la tècnica del Port Knocking amb la tècnica del SPA (Single Packet Authorization) i veurem les principals diferències entre tots dos mecanismes.

En primer lloc, veurem quines són les carències del PK que intenta solucionar la tècnica del SPA. A continuació, explicarem el funcionament general d'aquesta tècnica. Per acabar, veurem quina és la diferència substancial del funcionament d'ambdues tècniques.

6.1 Limitació del PK

En els apartats anteriors, hem vist que el Port Knocking és una tècnica que ens servirà per a obrir i tancar ports d'un servidor i a l'hora, fer que aquest procés sigui transparent per a possibles atacants. Per a l'intercanvi de missatges entre client PK i servidor PK hem vist que s'utilitzarà diferents camps de les capçaleres (a nivell de xarxa o transport) dels paquets enviats. Aquest mètode, que pot ser un avantatge per al funcionament de la tècnica, ja que ens permet "jugar" amb aquests camps per a l'intercanvi d'informació, és alhora una limitació. El fet de que utilitzem aquestes capçaleres (de longitud limitada) per a l'intercanvi de la informació limita la informació que podem codificar en elles.

Per tant, tenint en compte que els camps de dades dels paquets ens permeten enviar molta més informació que les capçaleres, què passaria si les utilitzéssim per a l'intercanvi d'informació entre client i servidor? Basant-se en aquesta premissa, sorgeix una nova tècnica, anomenada SPA (Single Packet Authorization).

6.2 SPA (Single Packet Authorization)

De la mateixa manera que la tècnica PK, el SPA es una tècnica que, mitjançant un sistema d'autenticació passiva i intercanviant les credencials correctes, un client podrà accedir a un determinat servei remot, a través d'un port que inicialment estava tancat. Per tant, tal i com passava amb el PK, aquesta tècnica tindrà un model client-servidor, amb una comunicació unidireccional del client cap al servidor.

Aquesta tècnica utilitzarà el camp de dades dels protocols per a l'intercanvi d'informació entre el client i el servidor. Tal i com hem vist amb la tècnica del Port Knocking, amb el SPA també podrem utilitzar diferents protocols per a l'intercanvi de la informació (ICMP, TCP i UDP). Tot i això, el fet d'utilitzar el camp de càrrega de dades augmenta la dificultat de la implementació. Per exemple, hi ha paquets ICMP que no disposen d'aquest camp, per tant, no serà possible utilitzar-los per a l'intercanvi de la informació.

Per tant, la limitació que tindrà SPA a l'hora d'intercanviar la informació serà la MTU, per tal d'evitar fragmentacions en els paquets i així, possibles problemes en la recepció dels mateixos per part del servidor.

6.3 SPA (Single Packet Authorization) vs PK (Port Knocking)

La principal diferència entre totes dues tècniques és la utilització de les capçaleres dels paquets, en el cas del PK o la utilització del camp de càrrega de dades, en el cas del SPA. A partir d'aquesta diferència en la implementació de l'intercanvi de missatges entre client i servidor, en sorgiran vàries:

SPA permet codificar més informació en un sol paquet ja que el PK estarà limitat per les capçaleres. Per exemple, podem autoritzar més d'una direcció IP, obrir o tancar més d'un port, implementar diferents mètodes de protecció a atacs,...

La implementació de SPA serà sensiblement més senzilla degut a la simplicitat a l'hora d'intercanviar informació entre client i servidor.

SPA només enviarà un missatge amb tota la informació mentre que PK haurà d'enviar una seqüència de missatges. D'aquesta manera, a part d'evitar possibles problemes en la recepció dels paquets, el temps de resposta serà millor en el primer cas.

Al contrari que en PK, els paquets SPA tindran normalment la mateixa longitud. D'aquesta manera, seran fàcilment distingibles a la resta del tràfic.

6.4 Conclusió

Com hem explicat, ambdues tècniques tenen un funcionament molt similar i es diferencien principalment en "com" el client i el servidor intercanvien aquesta informació. A arrel d'aquesta diferència, sorgeixen multitud de diferències entre la implementació d'una tècnica i l'altra. Per tant, tot i que aparentment el SPA pugui semblar una evolució de la tècnica del PK, en alguns casos serà recomanable l'ús d'una i, en d'altres, l'ús de l'altra.

Podem afirmar doncs que totes dues tècniques són vàlides i, tot i que SPA sembla solucionar alguna de les limitacions del PK, haurem d'estudiar molt bé l'escenari per a valorar quina de les dues tècniques és més adient.

07 Vulnerabilitats i atacs

Com hem comentat a l'apartat 3.4 – Propietats d'un sistema PK, el sistema PK haurà d'acomplir la propietat d'"ocultació", és a dir, que els paquets que s'intercanviïn clients PK i servidor PK hauran de ser indistingibles per a usuaris aliens a aquest procés. Tot i aquest fet, com veurem a continuació el sistema no serà completament invulnerable.

A l'apartat anterior, hem vist com hi ha diferents tipus d'implementació d'un sistema PK. Depenent del protocol sobre el qual es treballi, ens podem trobar amb possibles atacs i possibles vulnerabilitats, en les que un atacant podria comprometre el tràfic, entre altres coses, que s'envien clients i servidor. Així doncs, el servidor PK podria identificar un atacant com si aquest fos un client PK autoritzat.

Així doncs, depenent de la tècnica que s'utilitzi per a la transmissió dels missatges parlarem d'uns problemes/atacs possibles o d'uns altres. Per exemple, no serà el mateix una pèrdua dels missatges utilitzant UDP, on no hi ha control de si el paquet és rebut correctament, que si s'utilitza TCP, on el paquet serà retransmès fins a ser rebut correctament.

A continuació, comentarem alguns dels atacs o problemes amb els quals es podria comprometre la seguretat d'un sistema PK.

7.1 Possibles problemes

Ja hem comentat que la implementació del sistema PK dependrà en mesura de les necessitats que tingui l'administrador. Depenent de quins siguin els protocols que utilitzi per ala implementació del sistema PK, poden aparèixer certs problemes intrínsecs de cada un d'aquests protocols. A continuació en veurem alguns d'ells.

7.1.1 Pèrdua de paquets

Un dels principals problemes que pot afectar a un sistema de comunicació és el de la pèrdua de paquets. Tanmateix, aquest problema pot afectar a qualsevol sistema de comunicacions independentment de les tècniques de transmissió que s'utilitzin. Tot i que es tracta d'un problema de difícil solució, és cert que podem aplicar certs mecanismes de seguretat per a evitar que aquest problema afecti a la comunicació.

Entre alguns dels mecanismes que es poden utilitzar, està el d'afegir certa redundància als paquets. D'aquesta manera, un paquet perdut podria recuperar-se amb parts d'altres paquets enviats i rebuts durant la transmissió. Tot i això, aquest mecanisme implicaria augmentar el tamany dels paquets enviats, fet poc recomanable, ja que com hem comentat en punts anteriors, el fet de modificar el tamany dels missatges no és una bona opció degut a la reduïda capacitat de transmissió dels sistemes PK.

Com ja es sabut hi ha mecanismes a la capa de transport que ens permeten "combatre" la pèrdua de paquets. Utilitzant el protocol UDP, aquest no ens proporciona cap tipus de mecanisme per a confirmar que un paquet ha estat rebut correctament per part del receptor i, en cas de desitjar-ho, ho hem d'implementar en capes superiors. Aquest no és el cas de TCP. Aquest protocol ens proporciona control en la transmissió fins al punt que implementa mecanismes per a confirmar si un paquet ha estat rebut correctament i, en el cas de no ser així, implementa mecanismes per a la retransmissió del mateix.

Per tant, de cara al problema de la pèrdua de paquets, serà interessant implementar el nostre sistema PK sobre el protocol TCP. Hi ha una tècnica anomenada *Spread Spectrum TCP* [1], en la que un client podrà autenticar-se contra el servidor, tot i que part de la informació no arribi aquest últim.

7.1.2 Recepció desordenada

D'igual manera que el problema comentat anteriorment, aquesta problemàtica pot afectar a qualsevol sistema de comunicació, independentment de la tècnica de transmissió que s'utilitzi.

Imaginem que el client necessita enviar el missatge M al Servidor. Per a fer-ho ha de fragmentar-lo en 3 missatges (M_1, M_2 i M_3), que envia de manera ordenada cap al servidor. Però degut a la variació del canal, el servidor els rep de manera desordenada (M_2, M_1 i M_3). El servidor que utilitzi el protocol UDP, per exemple, no serà capaç d'interpretar-los correctament. De la mateixa manera que en el cas anterior, TCP proveeix un mecanisme de seguretat que assegura que el servidor serà capaç d'interpretar correctament aquests missatges.

A diferència que en el cas de la pèrdua de paquets, on TCP ens proveïa la solució al problema, en aquest cas la finestra TCP, que ens soluciona el problema de la recepció desordenada, no serà útil. Això és degut a que els sistemes de detecció de paquets en la recepció del servidor PK no depenen de la finestra TCP, sinó que accedeixen a la xarxa de manera directa. Així doncs, haurà de ser el propi sistema PK qui implementi algun mecanisme per a solucionar la recepció desordenada de paquets.

Alguna de les solucions que es poden implementar, entre d'altres, està la numeració de paquets, utilitzant algun camp de la capçalera. Una altra manera, seria la d'introduir cert retard entre la transmissió d'un paquet i l'altre, assegurant la correcta recepció del primer paquet abans d'enviar el següent i evitant així la variació de retard entre l'un i l'altre.

7.2 Vulnerabilitats i atacs

Hem vist que depenent del protocol que utilitzem a l'hora d'implementar el sistema PK, ens podem trobar en alguns casos amb certs problemes de xarxa que en d'altres poden ser solucionats amb mecanismes propis del protocol.

A continuació, veurem que el sistema PK també pot ser vulnerable a certs atacs que poden patir alguns protocols dels que utilitzem i que, en d'altres, aquesta vulnerabilitat pot ser solucionada mitjançant mecanismes de seguretat.

7.2.1 Atacs per repetició

Imaginem que un atacant està capturant paquets a la xarxa mentre un client PK legítim intenta autenticar-se contra el servidor PK i es fa amb aquests paquets. Si la informació transmesa no incorpora cap mecanisme de seguretat que permeti saber amb exactitud el moment en que ha estat transmès el paquet, l'atacant només hauria de suplantar la direcció IP del client PK i retransmetre els paquets d'autenticació. D'aquesta manera, el servidor PK considera que el client PK legítim està intentant realitzar una autenticació i realitzaria l'acció oportuna (la que té configurada i la mateixa que ha realitzat en l'intent legítim).

Per a evitar aquest tipus d'atacs, necessitem afegir algun tipus d'informació dins del missatge que ens permeti diferenciar peticions generades recentment i peticions antigues. Aquesta alternativa requereix que tant clients PK com Servidor PK tinguin uns rellotges de temps perfectament sincronitzats. D'altra manera, el servidor podria descartar intents d'autenticació legítims, per tractar-los de obsolets o caducats.

Una altra possible solució a aquest tipus d'atacs és el d'afegir alguna marca d'un sol ús, és a dir, un camp únic per missatge. Així doncs, el servidor PK comprovarà si el valor d'aquest camp ha estat ja utilitzat amb anterioritat i, de ser així, descartaria l'autenticació. Aquesta opció requereix únicament l'intercanvi d'aquest paràmetre

extra i que el servidor PK sigui capaç d'emmagatzemar els valors del mateix per tal de poder comprovar les noves peticions.

7.2.2 Enverinament

En l'apartat 05 – Variants existents, hem vist que hi ha moltes maneres d'implementar un sistema PK i que podem afegir diferents nivells de seguretat en els missatges que s'intercanvien clients PK i servidor PK. En la primera opció que hem comentat (missatges simples), on no es realitza cap tipus de comprovació d'autenticitat ni d'integritat, es relativament simple per a un atacant generar un missatge arbitrari amb l'objectiu de ser igual a un altre que impliqui l'autenticació amb el servidor PK i, que aquest últim n'actuï en conseqüència.

Llavors, ens podríem trobar en la situació en la que un atacant genera un missatge aleatori que és igual a un missatge que comporti que un Servidor PK obri un port i, per tant, doni accés il·legítim a un atacant per a un servei determinat.

Si volguéssim protegir el nostre sistema PK d'aquest tipus d'atacs hauríem d'afegir, com ja hem comentat, més mecanismes de seguretat (integritat i autenticitat). Tot i que augmenta considerablement la dificultat de la implementació del sistema, pot estalviar futurs mal de caps derivats d'atacs al nostre sistema.

08 Implementacions existents

En apartats anteriors, hem vist en què consisteix la tècnica del Port Knocking, quin és el seu funcionament i per a què serveix. En aquest punt, veurem algunes de les implementacions més populars i veurem el seu funcionament.

Com ja hem comentat amb anterioritat, la tècnica del Port Knocking és independent al Sistema Operatiu i al llenguatge de programació. Això ens obre un ventall immens de possibilitats a l'hora de trobar implementacions de la tècnica. A més a més, ja hem vist que es pot implementar sobre diferents protocols de la capa de transport (TCP, UDP) i aprofitant les possibilitats que ens ofereixen cada un d'aquests. Cada programador desenvoluparà la seva aplicació en funció de les seves necessitats i requeriments. Així doncs, ens podrem trobar amb implementacions molt simples que implementen la tècnica de manera molt senzilla o aplicacions que aprofiten mecanismes de control i de seguretat i que impliquen una implementació molt més complexa.

Per a fer-nos una idea de la quantitat d'aplicacions que implementen la tècnica del PK que podem trobar-nos a la xarxa, podem donar un cop d'ull al web http://www.portknocking.org/view/implementations. Aquí podem trobarem moltes aplicacions programades en diferents llenguatges de programació (Perl, JAVA, C, Python,...) i compatibles amb diferents Sistemes Operatius (Unix, Windows, MAC OS, IOS,...). Ens podem trobar inclús, amb aplicacions programades amb JAVA i altres llenguatges, i que són independents del S.O. i per tant, compatibles a la majoria. Inclús, aplicacions desenvolupades per a S.O. routers (com és el cas del OpenWRT).

A continuació, comentarem alguna de les aplicacions que ens podem trobar, analitzant el seu codi font i veient com funciona.

8.1 Knockd

8.1.1 Descripció

Autor: Judd Vinet <jvinet@zeroflux.org>

Llenguatge: Python

S.O. compatible: Unix, Windows, MacOS, Iphone (per al client)

Es tracta d'una aplicació que ha estat en constant desenvolupament i amb el pas del temps ha anat afegint opcions i fent-se compatible per a nombrosos entorns.

És una de les aplicacions més reconegudes quan es parla de la tècnica del Port Knocking.

El seu funcionament es basa en que un cop el Servidor PK rep una seqüència determinada i cas d'interpretar-la com a correcta, executa una comanda que li ha estat configurada prèviament.

Podem configurar el tancament del port de les dues maneres que hem comentat al punt *4.4 Finalització de la comunicació*, és a dir, tant per time-out (la connexió es tanca passat un temps que ha estat configurat al Servidor PK) o tancar la connexió utilitzant el mateix procés que a l'hora d'obrir el port, mitjançant la tècnica del PK.

8.1.2 Configuració

Servidor PK

Aquesta aplicació no es dedica a capturar els paquets entrants al Servidor sinó que s'encarregarà d'interpretar els fitxer de log que l'administrador del sistema li configuri.

En la configuració del Servidor PK, configurarem la ruta del fitxer de log que volem consultar i li configurarem els paràmetres del sistema:

Seqüència de ports. (Poden ser ports TCP o ports UDP)

Time out de les peticions.

Flags TCP. (Si volem utilitzar un determinat tipus de paquet TCP i ignorar-ne la resta. Per exemple: fin, syn, rst, psh, ack, urg)

Comanda que s'aplicarà un cop el servidor PK hagi rebut i interpretat correctament la seqüència enviada per part del client PK (per exemple, obrir un port en concret).

Opcionalment, també podem configurar el time out que s'aplicarà per a tancar la connexió i la comanda corresponent per a tancar el port.

8.2 KnockKnock

8.2.1 Descripció

Autor: Moxie Marlinspike (http://www.thoughtcrime.org/software/knockknock/)

Llenguatge: Python

S.O. compatible: Unix

Es tracta d'una aplicació molt completa, programada en python i que ha estat en constant actualització amb el pas del temps, fins al punt d'implementar mecanismes de seguretat per a evitar possibles atacs i vulnerabilitats.

Objectius de l'aplicació, segons l'autor:

Requeriments mínims. Per tant, es tracta d'una petita aplicació programada en Python.

No utilitza libpcap i que, per tant, no analitza cada paquet entrant al servidor. En el seu lloc, consulta fitxer de log.

No està basat en la implementació de UDP.

La interpretació dels paquets intercanviats entre el client i el servidor no ha de ser trivial per a un atacant que els pugui capturar.

El procés d'autenticació entre client i servidor s'ha de realitzar en un únic paquet.

Les capçaleres TCP són codificades de manera IND-CCA, per tal de ser "indistingibles" a la xarxa.

8.2.2 Configuració

Servidor

Hem de tenir en compte que el Servidor PK consultarà els missatges del fitxer kern.log[21], per tal d'identificar les peticions provinents d'un client PK. Per tant, haurem de configurar les regles al firewall per tal que les peticions correctes puguin ser correctament interpretades pel Servidor PK.

Per tant, haurem de configurar les regles del firewall per a que en quan es detecti una petició d'un client PK autoritzat, quedi reflexat al fitxer de log esmentat. Per exemple, creem una regla amb la seqüència de ports autoritzada de tal manera que en denegui el tràfic i ho reflexi al log amb el prefix "PK-REJECT".

Servidor PK

El programa funciona amb perfils. Cada perfil, constarà doncs d'un port, que serà el que es desitja obrir i serà associat a un usuari i màquina. Així doncs, per a crear un perfil, necessitarem el nom d'aquest i el port que desitgem obrir durant el procés PK.

Haurem de configurar el servidor PK per a que quan consulti el fitxer de log, busqui aquelles entrades que tenen el prefix que hem configurat a la regla del firewall. D'aquesta manera, el servidor PK sabrà en quin moment s'ha rebut una seqüència vàlida per a obrir un port determinat.

Un cop configurat el perfil per al servidor PK, ja podrem iniciar el programa a la espera de peticions entrants.

[21] El **kern.log** és un fitxer de logs on s'emmagatzemen els missatges del kernel generats per klogd.

8.2.3 Funcionament

El servidor inicia l'aplicació "knockknock-daemon". D'aquesta manera, s'inicia el Servidor PK.

Com ja hem comentat, aquesta aplicació no analitza els fitxers entrants al servidor sinó que analitza els logs d'aquest per tal d'analitzar els intents de connexió. En concret, analitza el fitxer kern.log, on s'emmagatzemen els logs del kernel, generats per klogd[22]. Així doncs, el servidor PK analitzarà els missatges del fitxer de log i en el moment de trobar un missatge que es correspongui amb una autenticació correcta, s'encarregarà d'obrir el port que li ha estat configurat.

Així doncs, ja tenim les regles del firewall configurades i el servidor PK configurat i executant-se. El client PK ja podrà enviar l'intent d'autenticació per tal d'obrir el port desitjat.

Un cop el servidor PK interpreti correctament l'intent de connexió d'un client PK, obrirà el port desitjat.

[22] Kernel Logging Daemon: Aplicació de Linux que escolta els missatges del kernel i és la responsable de procesar i prioritzar els missatges del S.O.

8.3 Altres

Hem vist com funcionen les aplicacions 'Knockd' i 'KnockKnock', però com ja hem comentat, existeixen moltes implementacions de la tècnica que s'adapten a diferents entorns i necessitats. Aquests en són alguns dels exemples amb els quals ens podem trobar:

Nom	Llenguatge	Plataforma	Descripció
Barricade	C	UNIX	Es tracta d'una implementació molt senzilla que obre un port determinat en rebre un paquet especial 'icmp echo'.
cd00r	C	UNIX	Es tracta d'una prova de concepte on el servidor PK analitza el tràfic en busca d'una seqüència correcta en qualsevol tipus de paquets IP.
cryptknock	C	UNIX	Implementació de la tècnica que, a diferència dels anteriors, utilitza una cadena de caràcters encriptada amb RC4 per a l'autenticació. El procés utilitza 3 paquets UDP.
KnockOnD	iOS	iPhone	Client PK per a executar-se des d'un terminal iOS, compatible amb el servidor Knockd.
wknock	C	Linux OpenWRT routers	Es tracta d'una prova de concepte que utilitza la tècnica del PK per a ocultar/mostrar WLANS configurades al router.

8.4 Cas pràctic

Imaginem que tenim un servidor amb un servei SSH executant-se, però no volem que romangui amb el port obert constantment, per tal de no exposar a un possible compromís la seguretat del servidor. Necessitarem utilitzar la tècnica del Port Knocking per tal de que, mitjançant la seqüència correcta, el servidor PK obri el port i puguem accedir-hi.

Per al desenvolupament de l'exemple, utilitzarem les següents màquines:

Servidor

O.S.: Linux Backtrack 5R2

Servei: SSH (port 22)

Firewall: Iptables amb tot el tràfic acceptat, excepte cap al port 22 que estarà tot denegat per defecte.

Servidor PK

Knokd v0.5

Client

O.S.: MacOS 10.7

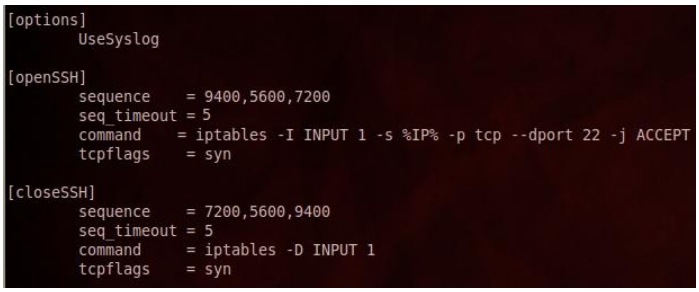
Client PK

Knock v0.5

8.4.1 Configuració

Servidor PK

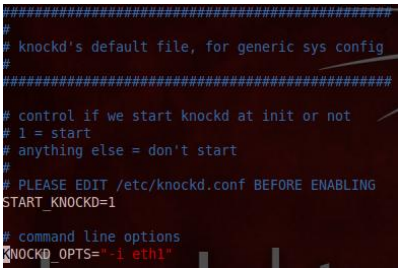
Un cop instal·lat el servidor PK en el servidor on desitgem obrir el port, haurem de procedir a la configuració. Per això, editarem el fitxer de configuració del servidor PK (/etc/knock.conf):



Com podem observar a la imatge, utilitzarem el Syslog per a la informació. Per a obrir el port, utilitzarem la seqüència indicada (9400, 5600, 7200). En rebre aquesta seqüència, el servidor PK executarà la comanda indicada. Aquesta comanda s'encarregarà d'afegir una regla permetent el tràfic al port 22 des de la IP del client PK i la col·locarà per sobre de la ja existent per tal de que faci un 'match' abans.

Per a tancar el port, el funcionament serà similar a l' anteriorment descrit. En aquest cas, la seqüència serà la inversa (7200,5600,9400) i la comanda executada, esborrarà la regla que abans havíem creat per a permetre el tràfic del port 22 des del client PK. D'aquesta manera, tornarem a bloquejar tot el tràfic cap al port 22.

A continuació, configurem la targeta per la qual anirà el tràfic d'entrada i activarem el servidor PK:



Procedirem a executar el servidor PK:

```
root@bt:~/Desktop/knock-0.5# /etc/init.d/knockd start
 * Starting Port-knock daemon knockd                                    [ OK ]
```

8.4.2 Execució

Abans de fer res, comprovarem l'estat del port 22 del servidor, des del client.

```
LuisPRO:Desktop Lluis$ nmap 192.168.0.108

Starting Nmap 5.51 ( http://nmap.org ) at 2013-01-13 00:38 CET
Nmap scan report for 192.168.0.108
Host is up (0.00020s latency).
Not shown: 999 closed ports
PORT   STATE    SERVICE
22/tcp filtered ssh
```

Com podem observar, el port està filtrat per firewall i, per tant, no podrem accedir al servei SSH.

A continuació, executem el client PK amb la seqüència correcta, per tal de que el servidor PK obri el port 22 i el client hi pugui accedir:

```
LuisPRO:Desktop Lluis$ ./knock 192.168.0.108 9400 5600 7200
```

Si ens fixem en el servidor, veurem com el servidor PK processa la petició i realitza l'acció configurada. Aquesta es la sortida del syslog durant la petició del client PK:

```
Jan 13 00:37:52 bt knockd: 192.168.0.107: openSSH: Stage 1
Jan 13 00:37:52 bt knockd: 192.168.0.107: openSSH: Stage 2
Jan 13 00:37:52 bt knockd: 192.168.0.107: openSSH: Stage 3
Jan 13 00:37:52 bt knockd: 192.168.0.107: openSSH: OPEN SESAME
Jan 13 00:37:52 bt knockd: openSSH: running command: iptables -I INPUT 1 -s 192.168.0.107 -p tcp --dpor
t 22 -j ACCEPT
```

Observem que el servidor PK ha detectat una seqüència correcta i ha executat la comanda indicada. Si observem les regles del firewall, comprovarem que ha introduït la regla que permet el tràfic des del client PK al port 22:

```
root@bt:~# iptables -n -L -v --line
Chain INPUT (policy ACCEPT 12575 packets, 845K bytes)
num   pkts bytes target     prot opt in     out     source               destination
1       51  5085 ACCEPT     tcp  --  *      *       192.168.0.107        0.0.0.0/0           tcp dpt:22
2       87  5268 DROP       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:22
```

Així doncs, si ara realitzem les comprovacions oportunes des del client PK, comprovarem que el port ha estat obert i el client PK ja hi pot accedir:

```
LuisPRO:Desktop Lluis$ nmap 192.168.0.108

Starting Nmap 5.51 ( http://nmap.org ) at 2013-01-13 00:38 CET
Nmap scan report for 192.168.0.108
Host is up (0.00031s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 0.25 seconds


LuisPRO:Desktop Lluis$ ./knock 192.168.0.108 9400 5600 7200
LuisPRO:Desktop Lluis$ ssh root@192.168.0.108
root@192.168.0.108's password:
Linux bt 3.2.6 #1 SMP Fri Feb 17 10:34:20 EST 2012 x86_64 GNU/Linux
Last login: Sun Jan 13 00:32:13 2013 from 192.168.0.107
root@bt:~# exit
logout
Connection to 192.168.0.108 closed.
```

Per a tancar el port, realitzarem el mateix procés però amb la seqüència inversa, que és la que hem configurat al servidor PK per a tancar el port 22:

```
LuisPRO:Desktop Lluis$ ./knock 192.168.0.108 7200 5600 9400
LuisPRO:Desktop Lluis$ nmap 192.168.0.108

Starting Nmap 5.51 ( http://nmap.org ) at 2013-01-13 00:38 CET
Nmap scan report for 192.168.0.108
Host is up (0.00020s latency).
Not shown: 999 closed ports
PORT   STATE    SERVICE
22/tcp filtered ssh
```

Comprovem que des del client PK, un cop iniciada la comanda, el port 22 torna a estar inaccessible.

Veiem què ha passat al servidor PK i com aquest ha tancat el port:

```
Jan 13 00:38:21 bt knockd: 192.168.0.107: closeSSH: Stage 1
Jan 13 00:38:21 bt knockd: 192.168.0.107: closeSSH: Stage 2
Jan 13 00:38:21 bt knockd: 192.168.0.107: closeSSH: Stage 3
Jan 13 00:38:21 bt knockd: 192.168.0.107: closeSSH: OPEN SESAME
Jan 13 00:38:21 bt knockd: closeSSH: running command: iptables -D INPUT 1
```

Si consultem les regles de l'Iptables, comprovem que en rebre la seqüència, el servidor PK ha esborrat la regla que abans havíem creat i, per tant, el port 22 torna a ser inaccessible:

```
root@bt:~# iptables -n -L -v --line
Chain INPUT (policy ACCEPT 15718 packets, 1048K bytes)
num   pkts bytes target     prot opt in     out     source               destination
1       89  5396 DROP       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:22
```

Hem de tenir en compte que la implementació del servidor PK ha estat molt senzilla perquè no hem implementat cap mètode de seguretat. Aquest sistema és vulnerable a

possibles atacs, com ja hem comentat. Tot i això, com hem pogut observar, la simplicitat en la implementació pot ser una de les possibilitats que faci decantar-se a un administrador per a aquest mètode tan simple i que afegeix un grau més de seguretat.

09 Conclusió

Durant tota la memòria, hem vist què és la tècnica del Port Knocking, hem vist el seu funcionament, quin és el paper que hi juguen cadascun dels actors implicats, en quines situacions ens pot ser útil aquesta tècnica, alguns dels possibles problemes i vulnerabilitats que pot patir i finalment, hem vist algunes de les implementacions ja existents i un cas pràctic en el que la tècnica ens podrà ser molt útil.

Ja hem comentat que tot i ser una tècnica amb una implementació inicial molt senzilla, mica en mica ha anat evolucionant per a garantir la comunicació entre servidor PK i clients PK, evitar possibles atacs dirigits a aquesta comunicació i solucionar problemes de transmissió de paquets. El fet d'afegir alguns d'aquests mecanismes, complicarà la implementació del sistema però el farà més segur. Aquest compromís entre la seguretat del sistema i la dificultat en la implementació serà decisió de l'administrador del sistema i de la situació en la que es desitgi implementar la tècnica del Port Knocking.

Al principi de la memòria hem comentat que la tècnica del Port Knocking ha estat bastant criticada des del món de la seguretat informàtica i tractada com a una tècnica d'obscuritat per al nostre sistema. En alguns casos, s'ha arribat a afirmar que pot ser un problema per a la seguretat del nostre sistema. Això és degut, com ja hem vist, a la infinitat d'opcions que un administrador té a l'hora d'implementar la tècnica del PK i adaptar-la al seu sistema. Si bé és cert que amb una implementació senzilla i sense cap mecanisme de seguretat addicional pot tractar-se d'una tècnica molt perillosa depenent de les circumstàncies. Una tècnica PK mal implementada podria convertir-se en un forat per a la seguretat del sistema i comprometre'l, convertint-se en una porta per a possibles atacants. Per aquesta raó, és aconsellable la implementació de la tècnica amb mecanismes de seguretat i fiabilitat.

Hem vist que existeixen alternatives a la tècnica PK, com pot ser el SPA. Aquesta tècnica pot ser recomanable abans que la tècnica PK en alguns casos, ja que pot arribar

a solucionar alguns dels inconvenients que té un sistema PK. Tot i això, no sempre serà així i, per tant, en altres ocasions l'opció d'implementar la tècnica PK serà recomanable per sobre de la opció del SPA.

En conclusió, l'administrador haurà d'analitzar molt bé el seu sistema per tal de decidir, en el cas de necessitar la implementació de la tècnica del Port Knocking, quines seran les opcions a escollir. En el cas d'ajustar correctament la tècnica a les necessitats del sistema, podem parlar d'un mecanisme segur que afegirà un grau més de seguretat al sistema.

09 Bibliografia

[1] Computer Security Institute, About the Computer Security Institute (CSI) <http://gocsi.com/about>

[2] W. Richard Stevens (1994), TCP/IP Illustrated Volume 1. Addison-Wesley Professional Computing Series.

[3] Bradley, T. (2004) Port Knocking. The New York Times, About, Inc. About.com. <http://netsecurity.about.com/cs/generalsecurity/a/aa032004.htm>

[4] Convery, S. (2004). Network Security Architectures. Cisco Press. Indianapolis, Estats Units.

[5] Deri, L. (2004). Improving Passive Packet Capture: Beyond Device Polling. Proceedings of SANE 2004. Holanda. <http://luca.ntop.org/Ring.pdf>

[6] Dobbertin, H. (1998). Cryptanalysis of MD4. German Information Security Agency. Alemanya. <http://www.cryptolounge.org/w/images/c/c2/Cryptanalysis of MD4.pdf>

[7] Gligor, V. (1993). A Guide to Understanding Covert Channel Analysis of Trusted Systems. National Computer Security Center. Maryland. Estats Units. <http://handle.dtic.mil/100.2/ADA276418>

[8] Murdoch, SJ. and Lewis, S. (2005). Embedding Covert Channels into TCP/IP. University of Cambridge, Computer Laboratory. Regne Unit. <http://www.cl.cam.ac.uk/~sjm217/papers/ih05coverttcp.pdf>

[9] Kerckchoffs, A. (1883) La cryptographie militaire. Journal des sciences militaires, Jan. 1883. <http://www.petitcolas.net/fabien/kerckhoffs/index.html>

[10] Mao, W. (2004). Modern cryptography. Theory and practice. Hewlett-Packard Company. Pearson Education. Prentice Hall Professional Technichal Reference. Estats Units.

[11] Mateti, P. (2000). Cryptography in Internet Security. Wright State University. Ohio. Estats Units. <http://www.cs.wright.edu/~pmateti/Courses/499/Cryptography/index.html>

[12] Mollin, R. (2007). An Introduction to Cryptography. 2nd edition. Chapman & Hall/CRC Press. United States.

[13] www.eumed.net. Curso de negocios por internet. [Presentació]
<www.eumed.net/cursecon/ecoinet/seguridad/hash.ppt>

[14] Firmas digitales y funciones de resumen y certificados.
<http://www.textoscientificos.com/redes/firewalls-distribuidos/soluciones-seguridad/firmas-funciones-resumen-certificados>

[15] Hou JC. (2004) 'Port Knocking'. Department of Computer Science, University of Illinois at Urbana Champaign. Estats Units.
<http://lion.cs.uiuc.edu/courses/cs397hou/lectures/PortKnocking.ppt>

[16] Internet Assigned Numbers Authority (2006) 'Port Numbers'.
<http://www.iana.org/assignments/port-numbers>

[17] Krivis S. (2004) 'Port Knocking: Helpful or Harmful? An Exploration of Modern Network Threats'. SANS Institute.
<http://www.giac.org/practical/GSEC/Stuart_Krivis_GSEC.pdf>

[18] Krzywinski M. (2003) 'Port Knocking: Network Authentication Across Closed Ports'. SysAdmin Magazine.

[19] Krzywinski M. (2003) 'Port Knocking'. Linux Journal, June 2003.
<http://www.linuxjournal.com/article/6811>

[20] Krzywinski M. (2004) 'A Critique of Port Knocking - Author's Response'.
<http://www.portknocking.org/view/about/critique/>

[21] Maddock B. (2004) 'Port Knocking: An Overview of Concepts, Issues and Implementations'. SANS Institute.
<http://www.giac.org/practical/GSEC/Ben_Maddock_GSEC.pdf>

[22] FX. (2000). Cd00r, packet coded backdoor.
<http://www.phenoelit-us.org/stu_/cd00r.c>

[23] Vinet, J. (2004). knockd - a port-knocking server.
<http://www.zeroux.org/projects/knock>

[24] Martín, Luis. 'Técnicas avanzadas de filtrado dinámico en sistemas cortafuegos.' (2010)

[35] Jaime Gutiérrez (2003), 'Protocolos criptográficos y seguridad en redes.' Universidad de Cantabria. Servicio de Publicaciones.

[36] Ahsan, K. (2002). 'Covert Channel Analysis and Data Hiding in TCP/IP. Department of Electrical and Computer Engineering'. University of Toronto. Canadà.
<http://gray-world.net/papers/ahsan02.pdf>

[37] Murdoch, SJ. and Lewis, S. (2005). 'Embedding Covert Channels into TCP/IP'. University of Cambridge, Computer Laboratory. Regne Unit. <http://www.cl.cam.ac.uk/~sjm217/papers/ih05coverttcp.pdf>

[38] Zalewski, M. (2001). 'Strange Attractors and TCP/IP Sequence Number Analysis'. BindView Corporation. Estats Units. <http://lcamtuf.coredump.cx/oldtcp/tcpseq/print.html>

[39] The Swiss Education and Research Network. (2002). 'Default TTL values in TCP/IP'.Suïssa. <http://www.map.meteoswiss.ch/map-doc/ftp-probleme.htm>

[40] Bellovin, SM. (1989). 'Security Problems in the TCP/IP Protocol Suite.' AT&T Bell Laboratories. Estats Units. <http://www.cs.columbia.edu/~smb/papers/ipext.pdf>

[41] http://maxid.com.ar (2011). 'Port Knocking con ICMP en MikroTik' <http://maxid.com.ar/port-knocking-con-icmp-en-mikrotik>

[42] Amir R. Khakpour, and Hakima Chaouchi, 'ESSTCP: Enhanced Spread-Spectrum TCP' (2007). Proc. of the 3rd International Workshop on Security in Systems and Networks (SSN'07). Estats Units.