Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Informàtica

Títol: Anàlisi del trànsit Wi-Fi basat en Deep Learning per a la detecció d'un atac

Document: Memòria

Alumne: Oliu Llorente Moragrega

Tutor: Lluís Fàbrega Soler
Departament: ATC
Àrea: ATC

Convocatòria (mes/any): Juny/2021

0. Taula de continguts

1. Introducció i objectius

1.1. Introducció

Les xarxes Wi-Fi són una popular tecnologia de xarxa sense fils basada en el conjunt d'estàndards IEEE 802.11, aquestes pateixen de manera constant atacs a la seguretat de diversos tipus (p.e., d'obtenció de la clau de xifrat, de desassociació, de suplantació del Punt d'Accés, etc.), que atempten contra la privacitat i integritat de la informació i contra la disponibilitat de la xarxa.

Per identificar aquests atacs (i així potser evitar-los) s'utilitzen sistemes de detecció d'intrusions (Intrusion Detection Systems o IDS), que monitoren el trànsit de xarxa i l'analitzen fent servir diverses tècniques (comparació amb patrons d'atacs coneguts, comparació amb un model que representi una situació "normal", etc.) que permeten detectar si s'està produint un atac, de manera que es pugui actuar en conseqüència.

Darrerament s'està explorant l'ús de tècniques d'intel·ligència artificial (concretament de Deep Learning (DL)) per millorar el rendiment dels sistemes de detecció d'intrusions, gràcies a la seva habilitat per aprendre, de manera automàtica, les característiques més determinants d'un conjunt de casos predefinits, i la posterior detecció i classificació de nous casos no coneguts.

1.2. Objectius

L'objectiu del projecte és dissenyar i implementar un programa que, basat en un dataset que contingui dades d'atacs Wi-Fi (AWID2) i una llibreria de Deep Learning (*fastai*), sigui capaç d'aprendre i detectar un tipus concret (ja definit) d'atac Wi-Fi. També s'hauran d'aplicar i consolidar coneixements de Xarxes i ampliar els coneixements (bàsics) d'intel·ligència artificial adquirits durant la carrera.

El projecte es realitzarà en col·laboració amb el grup de recerca "Broadband Network Control and Management" (BCDS) de l'UdG, el qual proporcionarà el hardware necessari per a dur-lo a terme. Concretament, del grup de recerca es realitzarà el projecte amb en Lluís Fàbrega Soler (tutor del TFG), en Miquel Farreras Casamort (estudiant de doctorand al grup) i en Pere Vilà Talleda.

Els passos a seguir per tal de dur a terme l'objectiu seran els següents:

- Investigació i aprenentatge sobre Deep Learning i més concretament sobre la llibreria *fastai* per a Python.
- Investigació i aprenentatge dels diferents atacs Wi-Fi.
- Selecció d'un atac Wi-Fi concret sobre el qual realitzar el projecte.
- Comprensió i filtratge del dataset AWID2 del qual s'obtindran les dades per entrenar el programa de detecció.
- Implementació del programa en Python que utilitzi la llibreria *fastai* per a crear un model que, mitjançant Deep Learning, aprengui del dataset.
- Proves d'eficiència i precisió del programa.
- Prova del programa en un escenari real realitzant l'atac en directe i capturant-ne el trànsit (objectiu opcional condicionat al temps).

1.3. Origen del projecte i motivacions personals

L'origen del projecte es basa en una proposta del tutor que consistia en fer servir la llibreria *fastai*, una manera senzilla i potent de fer un model de Deep Learning en un temps raonable per un TFG i fer servir el dataset d'AWID2, el qual és molt complet i amb molta documentació, per a realitzar el sistema de detecció.

A nivell personal m'interessa molt l'àmbit de les xarxes i de la tecnologia de la informació, i si té a veure amb la seguretat encara més. El grup de recerca de la UdG "Broadband Communications and Distributed Systems" (BCDS) estava buscant nous temes d'investigació dins el seu àmbit de gestió de xarxa (p.e., mecanismes de xarxa com el routing, robustesa, seguretat, grafs, etc.). Un dels temes que semblava interessant, era el d'aplicar tècniques d'intel·ligència artificial (concretament de Deep Learning) per a detectar un tipus concret d'atac Wi-Fi a través del trànsit de la xarxa. Per tant, quan va sorgir la possibilitat de fer el projecte sobre aquest tema, vaig trobar molt intrigant el fet de poder dissenyar un programa que "aprengués" sobre atacs Wi-Fi per tal de detectar-ne en el futur.

També m'agraden molt els temes relacionats amb la intel·ligència artificial (encara que no en tingui masses coneixements), i el fet de dissenyar un model que no depengui d'esquemes "predefinits", sinó que sigui capaç d'aprendre per ell mateix.

2. Estudi de viabilitat

2.1. Recursos necessaris

Per a dur a terme el treball adequadament no és necessari comptar amb equipament molt específic, només cal disposar de hardware amb suficient capacitat de computació per a poder entrenar un model d'intel·ligència artificial en un temps "acceptable".

Per la part de hardware s'utilitzarà un servidor del què ja disposa el grup de recerca BCDS localitzat a l'edifici P-IV de l'Escola Politècnica Superior de la Universitat de Girona al campus de Montilivi. Aquest servidor compta amb les característiques necessàries per a poder crear i entrenar un model de Deep Learning sense problemes.

Les característiques del servidor són les següents:

- **Processador**: Intel I7 2600k 3.8GHz (4 cores 8 threads)
- **Memòria RAM**: 16 GB RAM DDR3
- **Targeta gràfica**: nVidia GeForce GTX 1070 (8 GB GDDR5)
- **Emmagatzematge**: 1 TB HDD + 256 GB SSD
- **Sistema operatius**: Ubuntu Server 20.04 LTS

El servidor tindrà instal·lat Python 3 (entre altres coses) i comptarà amb el servei gratuït de JupyterHub, el qual permet tenir en un mateix servidor diferents usuaris, on cada un pot tenir-hi instal·lats els seus propis paquets de Python i les seves pròpies notebooks[1].

Els recursos de software seran els següents:

- **Un usuari amb accés al JupyterHub del servidor**: S'hi podrà accedir mitjançant la seva interfície web i s'hi podran pujar els datasets que es facin servir, a part de tenir-hi un notebook de Python amb el programa de detecció d'atacs implementat.

[1] Una jupyter notebook és una aplicació que permet crear i compartir documents web en format JSON amb un esquema de cel·les d'entrada i sortida. Les cel·les poden ser de codi, text en format markdown, formules matemàtiques i equacions (entre altres coses).

- **Llibreria de Deep Learning *fastai***: És una llibreria gratuïta que permet dissenyar i implementar models de Deep Learning a "alt nivell".
- **Dataset AWID2**: És el dataset d'on s'extrauran les dades per a entrenar i testejar el model. És un dataset molt complet realitzat per una universitat, consta de molta documentació i és utilitzat en molts projectes.

2.2. Recursos Humans

L'única despesa en aquest sentit seré jo, ja que sóc l'únic que estaré treballant en el projecte. El cost seria pagar el meu "sou", que consistiria a pagar les hores totals dedicades al projecte.

Agafem com a preu hora d'un enginyer informàtic 14 euros/hora.

Si per aquest projecte s'hi dediquen aproximadament unes 303 hores (a l'apartat 4 es veu d'on surt aquest valor) el preu total en recursos humans serà de 4.242 euros.

2.3. Viabilitat econòmica

Econòmicament el projecte valdrà la suma del cost dels recursos de hardware, els recursos de software i dels recursos humans:

- Els recursos de hardware no suposen cap cost extra, ja que els proporciona el grup de recerca i ja estaven comprats d'abans (no s'ha de comprar res nou).
- Dels recursos de software, la llibreria *fastai* és gratuïta i el dataset AWID2 també ho és, ja que només cal registrar-se amb el correu de la universitat per tal que el proporcionin.
- Per als recursos humans, tal com s'ha vist a l'apartat 2.2, el cost serà de 4.242 euros.

El cost total del projecte serà de 4.242 euros (ja que només els recursos humans tenen cost monetari).

2.4. Viabilitat tecnològica

Com s'ha mencionat anteriorment, el hardware del qual es disposa compta amb suficients recursos per a poder dur a terme el projecte (tant en termes de capacitat de computació com en termes de memòria i espai).

En relació al software, la llibreria *fastai* és prou potent i completa per a implementar un sistema de Deep Learning en prou temps perquè sigui viable (no requereix més temps del necessari per a dur a terme un TFG). El dataset d'AWID2 també en permet la viabilitat, ja que és molt complet, conté molts atacs representats i molts tipus de datasets diferents (tant d'entrenament com de validació).

Així doncs, podem concloure que el projecte és viable tant a nivell de hardware (ja que tindrem suficient capacitat de computació) com a nivell de software (ja que tant la llibreria utilitzada com el dataset permeten realitzar el projecte en un temps raonable).

3. Metodologia

Per a realitzar el projecte s'hauran de realitzar una sèrie de passos tant d'aprenentatge i recerca com d'implementació. La majoria d'aquests passos són en ordre seqüencial, però alguns es podran fer en paral·lel (pe., els relacionats amb investigar, encara que sigui sobre temes diferents).

L'ordre dels passos és el següent:

1. Aprendre, investigar i assolir conceptes bàsics de Deep Learning i de la llibreria *fastai*.
2. Paral·lelament al pas 1, investigar i aprendre sobre els diferents atacs Wi-Fi (d'entre els atacs contemplats al dataset AWID2).
3. Decidir sobre quin atac fer el sistema de detecció en funció de lo aprés al pas 2.
4. Comprendre, interpretar i filtrar el dataset AWID2.
5. Construir el sistema de detecció d'atacs (del tipus escollit al pas 3).
6. Provar els resultats i la precisió del sistema.
7. Provar el sistema de detecció d'atacs en un escenari de laboratori.
8. Documentació i redacció de la memòria.

4. Planificació

El projecte comença el 19 de febrer un cop aprovat el full de projecte, i s'entregarà el dia 11 de juny (com a molt tard).

S'intentarà dedicar unes 3 hores al dia de mitjana, entenent que hi haurà dies on s'hi dedicaran menys de 3 hores i dies on se n'hi dedicaran més.

El projecte es planificarà segons surt a la Figura 1

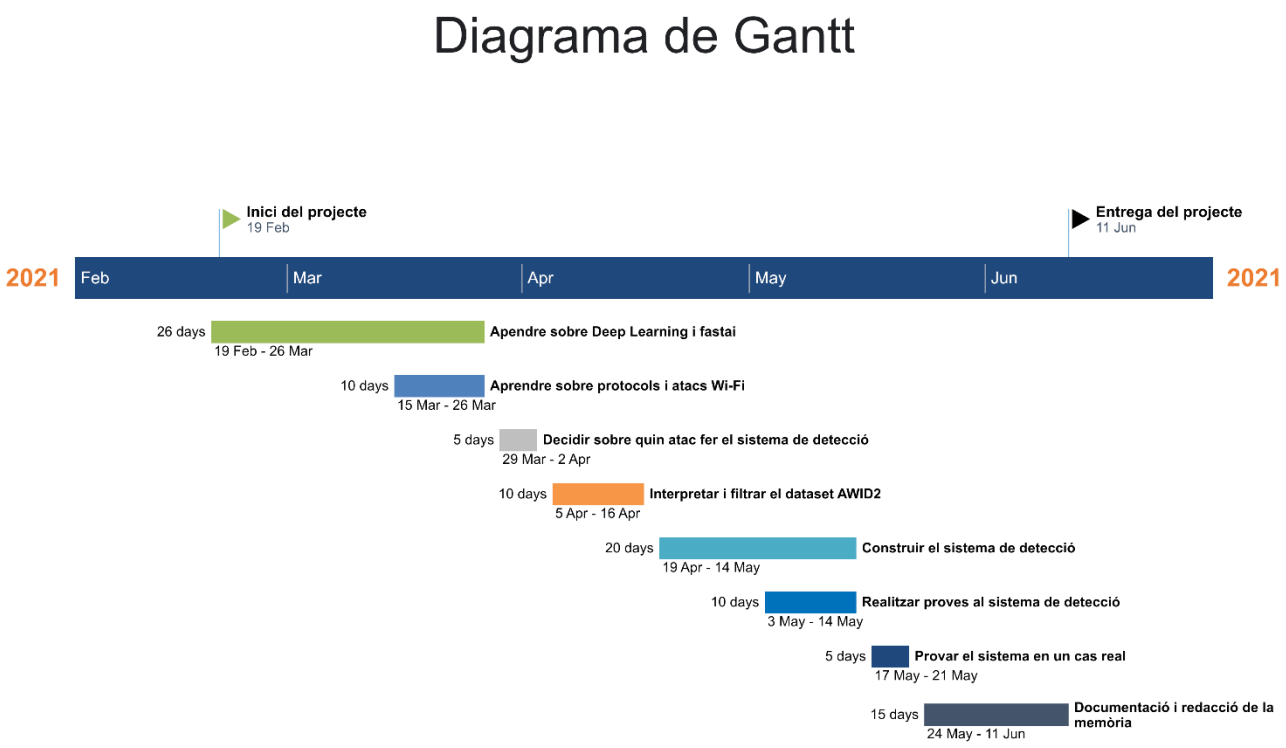


Figura 1: Planificació del projecte

En total es tardarà 101 dies a realitzar el projecte. Amb un total aproximat de 303 hores dedicades.

Cal indicar que el projecte s'ha planificat de manera que es facin reunions amb el tutor setmanalment (cada dilluns) des de l'inici del projecte fins a la seva defensa oral (el dia 23 de juny de 2021).

5. Marc de treball i conceptes previs

5.1. Intel·ligència artificial

La definició general d'intel·ligència artificial és "L'habilitat d'una màquina de poder presentar les mateixes capacitats que els éssers humans, com el raonament, l'aprenentatge, la creativitat, etc." [1].

En altres paraules, la intel·ligència artificial és un sistema informàtic que, a través d'unes dades (siguin definides prèviament o bé recopilades "in situ") és capaç de processar el que està passant, adaptar el seu comportament de manera dinàmica i actuar en conseqüència (sempre dins les funcions per les quals estigui dissenyat).

Aquest projecte es centrarà en utilitzar tècniques d'intel·ligència artificial que simulen l'aprenentatge, de manera que es pugui desenvolupar un sistema que sigui capaç d'aprendre per a poder predir una determinada situació (en aquest cas un tipus determinat d'atac Wi-Fi).

En el camp de la intel·ligència artificial s'utilitza molt l'aprenentatge de màquina (Machine Learning) per tal de simular el procés d'aprenentatge d'un humà, però s'estan començant a fer servir tècniques que van una mica més enllà, com les de Deep Learning (sobre les quals basarem el projecte).

5.1.1. Aprenentatge de màquina (Machine Learning)

L'aprenentatge de màquina és el camp de la intel·ligència artificial que permet a sistemes informàtics identificar patrons d'entre grans quantitats de dades per a poder-ne fer prediccions o classificacions. És a dir, són sistemes informàtics que a través de diferents tipus d'algoritmes, són capaços d'aprendre automàticament mitjançant l'experiència utilitzant grans quantitats de dades.

Quan es parla de les dades que reben aquests algoritmes es parla de datasets. Aquests datasets es poden utilitzar tant per entrenar les dades (els anomenats **datasets d'entrenament**), com per comprovar si el model (fent referència al programa que s'entrena) s'ha entrenat correctament amb dades que no ha vist prèviament (els anomenats **datasets de test**).

Una cosa important que s'ha de tenir en compte en l'aprenentatge de màquina és que no hi hagi l'anomenat "**overfitting**", que significa que el model no hagi "memoritzat" les dades en comptes d'aprendre d'elles.

Els algoritmes d'aprenentatge de màquina es poden dividir en dues categories principals [4]:

- **Aprenentatge supervisat**: Les dades d'entrenament inclouen tant "l'input" com els resultats esperats, de manera que els algoritmes saben quin resultat s'espera a l'hora d'entrenar, permetent que els algoritmes siguin ràpids i precisos. Tot i tenir accés als resultats esperats han de saber generalitzar, han de ser capaços d'obtenir resultats correctes a partir de dades noves.
- **Aprenentatge no supervisat**: A diferència de l'aprenentatge supervisat, el model no disposa dels resultats esperats per a realitzar l'entrenament. Es basa en classificar les dades basant-se únicament en les seves propietats estadístiques.

5.1.2. Deep Learning

Durant l'última dècada han estat prenent més força un subconjunt de tècniques del Machine Learning anomenades Deep Learning [5] (Figura 2), que consisteixen en fer servir la idea d'aprendre a partir de l'exemple.

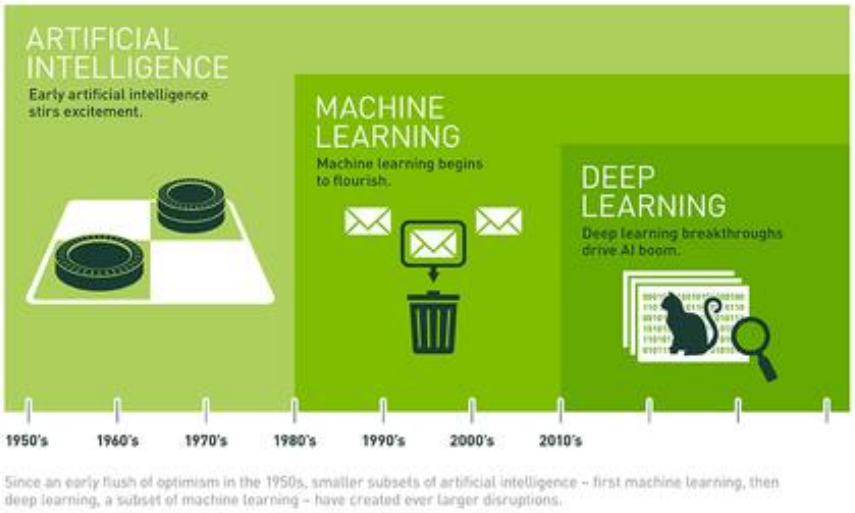


Figura 2: Evolució de la intel·ligència artificial [5]

El Deep Learning [5] consisteix en donar a l'ordinador un model que pugui avaluar en comptes de donar-li una llista de regles predefinida. Això permet que a mesura que passi el temps el model hagi sigut capaç de trobar certs patrons d'entre les dades, de manera que pugui resoldre de forma molt precisa el que se li demani.

De totes les tècniques que serveixen per implementar Deep Learning, una de les més utilitzades (i la que es farà servir en aquest projecte) és la que es basa en simular i implementar les anomenades **xarxes neuronals artificials** [6] (Artificial neural networks o ANN en forma abreviada) que, com el seu nom indica, intenten "emular" el funcionament del cervell humà.

Aquestes xarxes neuronals estan formades per un conjunt de "neurones" agrupades en diferents capes, connectades entre elles i que tenen una direcció concreta cap a on es propaguen les dades. Cada neurona (Figura 3) consta de diverses entrades (inputs) i una única sortida (output) que es pot enviar a altres neurones. Els diferents inputs poden ser tant els valors de les dades sobre les quals s'està treballant com els outputs d'altres neurones.

D'entre totes les neurones, hi ha les anomenades neurones de sortida (output neurons), les quals duen a terme la tasca per la qual s'ha implementat la xarxa neuronal (pe., classificar una imatge, predir un determinat valor, identificar un determinat element, etc.).

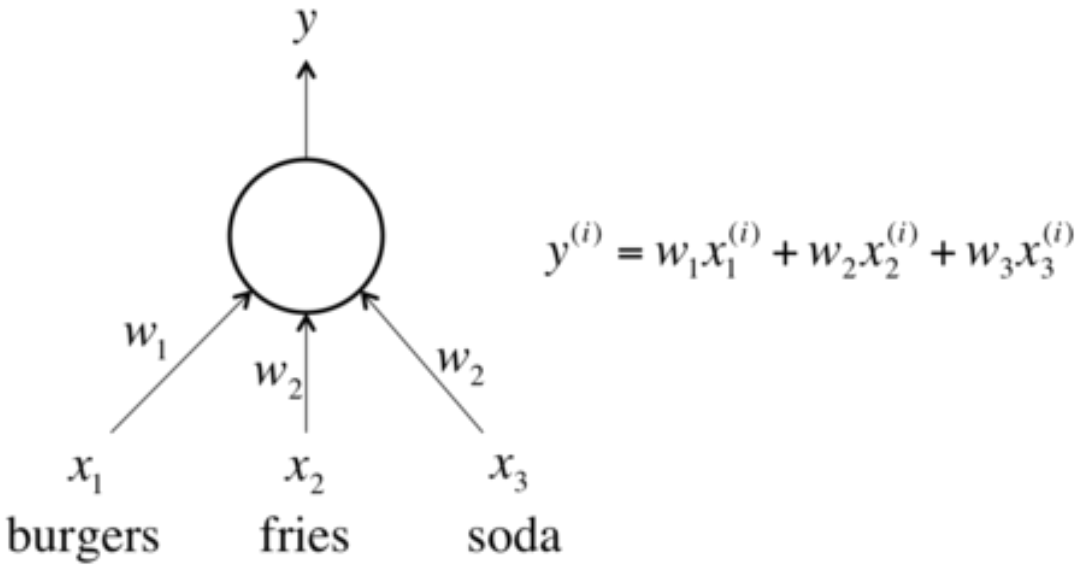


Figura 3: Imatge d'una neurona [6]

Cada una de les connexions de la xarxa consta de la sortida d'una neurona com a entrada (o una de les entrades) d'una altra neurona. Cada connexió té assignada un "pes", que representa la importància de la connexió dins la xarxa. D'aquí en surt la funció de propagació, que fa una suma de pesos de totes les connexions de la xarxa.

Aquestes neurones s'organitzen en múltiples capes (Figura 4), de manera que les neurones d'una capa es connecten únicament amb les de la capa immediatament anterior i/o les de la capa immediatament posterior. La capa que rep la informació de l'exterior (les dades d'entrada) s'anomena "input layer", i la capa on hi ha les neurones que tenen com a sortida el resultat s'anomena "output layer". Entre la "input layer" i la "output layer" hi pot haver (o no) altres capes, les quals s'anomenen "hidden layers".

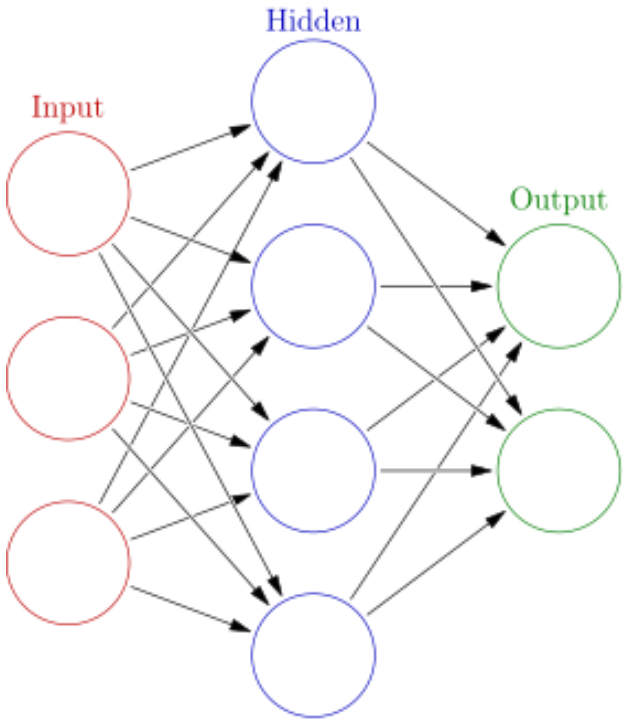


Figura 4: Estructura de les capes d'una xarxa neuronal

Un altre terme important és el de "**learning rate**" o ritme d'aprenentatge. Aquest terme fa referència al nombre de correccions que ha de fer el model per adaptar-se als errors. Això es tradueix en que un learning rate alt disminueix el temps d'entrenament a costa de perdre precisió mentre que un model amb un learning rate més baix tardarà més a entrenar-se, però serà més precís.

5.2. Fastai

Figura 5: Llibreria fastai

Per a implementar el model de Deep Learning, s'haurà d'utilitzar una llibreria que permeti dur a terme accions relacionades amb la intel·ligència artificial dins de Python. D'entre les diferents llibreries que hi ha en aquest àmbit, s'utilitzarà la llibreria de *fastai* (Figura 5) [11].

Fastai és una llibreria open source de Python que treballa per sobre de PyTorch (un dels frameworks de Deep Learning més utilitzats). El seu principal objectiu és intentar fer tot el procés de creació de xarxes neuronals tan fàcil com sigui possible però de manera ràpida i precisa. Volen fer que la IA sigui accessible a tothom, sense importar els coneixements i habilitats que es tinguin sobre aquest món. En altres paraules, *fastai* és una llibreria de Deep Learning feta per treballar a "molt alt nivell".

A més, és una llibreria que consta de molta documentació (pe. cursos, pàgines de documentació de les funcions, fòrums propis, etc.), cosa que permetrà trobar solucions a problemes futurs que puguin sorgir i també ajudes i guies a l'hora de desenvolupar el model.

La llibreria *fastai* permet aplicar Depp Learning en una gran varietat de tipus de projectes (Figura 6). Des de projectes de visió per computador (pe. classificadors de fotografies, prediccions en temps real capturant imatges de l'entorn, etc.), fins a projectes de dades en format tabular, com es farà en aquest projecte.

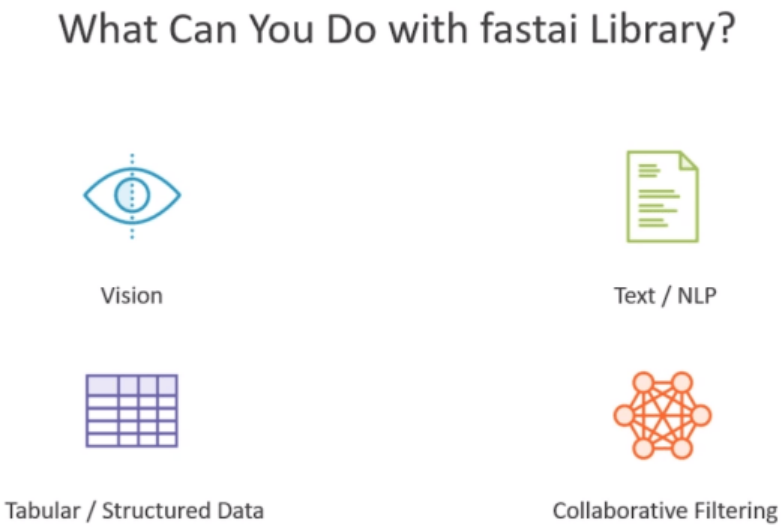


Figura 6: Àmbits d'us de fastai

5.3. Seguretat Wi-Fi

Abans de l'explicació, és necessari introduir una sèrie de conceptes pel que fa a la nomenclatura utilitzada més endavant. Una "**trama**" ("frame") Wi-Fi és equivalent a un paquet Wi-Fi, però en termes més tècnics últimament s'utilitza la paraula trama.

Quan es fa referència a un "**client**" d'una xarxa Wi-Fi, s'està fent referència a qualsevol dispositiu o estació que estigui connectat a la xarxa (o que vulgui associar-s'hi). Per últim, indicar que al fer referència als Access Points (Punts d'accés) tant es farà pel nom com per l'abreviació: "**AP**".

5.3.1. Trames en les connexions Wi-Fi

- **Trames de gestió (Figura 7) [7]**: Permeten als clients establir comunicació amb els AP i mantenir-ne la connexió, n'hi ha de varis tipus:
  - **Authentication**
  - **Deauthentication**
  - **Association request**
  - **Association response**
  - **Reassociation request**
  - **Reassotiation response**

- **Dissasociation**
- **Beacon**: Trames transmeses periòdicament per anunciar la presencia d'un AP i les seves capacitats.
- **Probe request**: Un tipus de petició especial enviada per un client demanant informació sobre un punt d'accés en concret o sobre tots els punts d'accés d'una xarxa.
- **Probe response:** Resposta a la probe request amb la informació demanada.

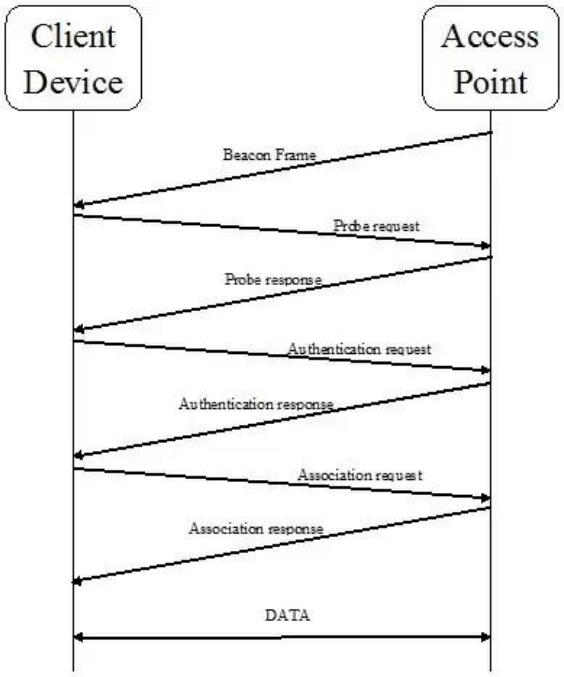


Figura 7: Seqüència temporal d'una connexió a una xrxa Wi-Fi

- **Trames de control [7]**: Coordinen l'accés al canal sense fils i participen en l'enviament de les trames de dades (es veuen més avall) entre els AP i els clients. N'hi ha de diversos tipus:
  - **Request to send (RTS)**: És el primer pas del mecanisme (opcional) de "handshake", el qual es basa en que el client envïi una trama RTS on demani permís per ocupar el canal abans de transmetre informació "real".
  - **Clear to send**: Resposta a l'RTS indicant que el canal està lliure i que es pot procedir amb l'enviament.
  - **Acknowledgement**
  - **Power Save (PS) Poll**

- **Trames de dades [7]**: Són les que transporten la informació com a tal (que prové d'altres capes). N'hi ha de diferents tipus en funció de si la informació s'envia en un servei basat en contencions, si porten informació addicional i/o si tenen temes de qualitat de servei (QoS).

Les trames s'estructuren en capçalera, el cos de la trama i la "frame check sequence" (FCS) amb llargada de 4 bytes. Les dades del cos (l'únic camp amb longitud variable) de la trama acostumen a estar encriptades, l'encriptació tant del cos com de la capçalera es basa en un algoritme de xifrat conegut com a CRC-32. També en relació a la capçalera, aquesta ocupa 30 bytes i consta de 7 camps diferents. Les trames de gestió tenen el cos amb longitud fixe o amb paràmetres variables que ja estiguin etiquetats. Per altra banda, les trames de control no tenen cos i tenen la capçalera més petita.

5.3.2. Seguretat WEP

Va ser el primer mecanisme de seguretat del protocol 802.11 (1999) [7], amb l'objectiu de donar confidencialitat. Tot i això, aquest protocol no va aconseguir el nivell de confidencialitat desitjat ja que era susceptible a un gran nombre d'atacs.

- **Autorització:** WEP té dos mètodes d'autenticació. Un és "Open System" (l'autenticació es completa amb 2 missatges), on el client no ha d'aportar cap tipus de credencials per a establir connexió ja que la seguretat (habitualment) es basa en una Whitelist d'adreces MAC.

  L'altre és el mètode de clau compartida, el qual requereix 4 missatges per tal que el client es pugui connectar a la xarxa: El client envia una petició d'autenticació que conté tant l'@MAC del client com la de l'AP, després l'AP respon amb un missatge que conté un nombre aleatori de 128 bits, a continuació el client envia un missatge que conté el nombre anterior encriptat amb la clau WEP compartida, i finalment l'AP desencripta el missatge anterior utilitzant la clau compartida i comprova si el nombre desencriptat correspon amb el generat en el segon pas. Si és el cas, permetrà l'accés del client a la xarxa, encara que, tant si l'accepta com si no, enviarà un missatge "d'authentication response" indicant si se li permet o no l'entrada a la xarxa al client.

- **Encriptació del trànsit:** El protocol WEP depèn de l'algoritme RC4 per la confidencialitat i de l'algoritme CRC-32 per a la integritat del missatge [7]. La confidencialitat en aquest protocol es basa en una clau estàtica coneguda com a "root key". El protocol WEP consta de 2 versions en funció de la mida de la clau: WEP-40 i WEP-104, on el nombre que acompanya el protocol és la mida de la clau. En ambdós casos la clau no s'utilitza directament per a encriptar paquets però és la base per a la generació d'una clau de sessió. En el procés d'encriptació (on només s'encripten les trames de dades) es duen a terme els següents passos: Primer es genera un vector d'inicialització (IV)[2] de 24 bits, després es concatena la "root key" amb aquest vector i finalment s'agafa una seqüència de claus generada a partir de la "root key" i es fa una XOR amb el text en clar del paquet i el seu valor de CRC-32.

5.3.3. Seguretat WPA

Va ser el protocol de transició entre WEP i WPA2 [7], ja que WPA es basava en solucions de seguretat de tercers per a millorar i resoldre els problemes de seguretat de WEP. Aquest protocol depèn de servidors centrals d'autenticació com RADIUS per a autenticar els usuaris.

5.3.4. Seguretat WPA2

Es basa en el protocol 802.11 i té uns sistemes de construcció de clau i de confidencialitat i integritat del trànsit més complexos [7]:

- **Construcció de la clau**: En WPA2 totes les claus provenen d'una clau única que està a sobre de tot de la cadena. Hi ha 2 tipus de claus, les quals depenen del mètode d'autenticació ja que, si aquest és basa en una clau prèviament compartida, la clau "base" serà simplement la clau precompartida (PSK). Per altra banda si el mètode utilitzat es basa en el framework 802.1X, la clau "base" serà una clau anomenada "Master session key" (MSK).

  Qualsevol de les 2 claus es podran utilitzar per al que es coneix com a "primary keying material", que en WPA2 fa referència a la "Pairwise Master Key" (PMK). En el cas de clau precompartida, la PMK serà la PSK, mentre

[2] El vector d'inicialització (IV) [8] és un bloc de bits necessari per a permetre diferents tipus de xifratge. El tamany d'aquest vector depèn tant de l'algoritme de xifratge com del protocol criptogràfic. L'objectiu d'aquest vector és el de solucionar problemes de seguretat d'altres mètodes.

que si és l'altre mètode, la PMK serà una porció de la MSK. La clau PMK mai s'usa per a encriptació directament, sinó que contribueix a la generació de claus (amb vida més curta). Durant el procés d'autenticació es generen 2 claus específiques per a connexió AP-client, són la "Pairwise Transistent key" (PTK) i la "Group Transistent Key" (GTK), les quals venen de l'MSK o de la PMK. Finalment la PTK es divideix en 5 subclaus i la GTK es divideix en 2 subclaus. Cada una d'aquestes subclaus té la seva funció específica.

- **Confidencialitat i integritat del trànsit**: WPA2 suporta 3 protocols en aquest aspecte: Temporal Key integrity protocol (TKIP), Counter-mode/Cipher Block Chaining Message Authentication Code Protocol (CCMP) i "Wireless Robust Authentication Protocol" (WRAP). El protocol TKIP es basa en l'RC4, el protocol WRAP es basa en "l'Offset codebook" (OCB) mode derivat de l'AES (Advanced Encryptation Standard) el qual es considera molt més segur. Finalment el CCMP es basa en el mode CCM de l'algoritme AES.

Per aquest projecte no cal entrar en detall en que fan molts dels algoritmes d'encriptació però si que està bé veure l'evolució dels protocols de seguretat i explicar les diferències i millores d'uns respecte els altres.

5.4. El dataset

En el món del Deep Learning (i de la intel·ligència artificial en general) és tan important dissenyar un bon model d'entrenament com seleccionar bé les dades que s'utilitzaran per entrenar-lo i testejar-lo, ja que si aquestes no són prou completes o representatives poden provocar que el model s'entreni malament o que directament no funcioni.

En aquest projecte s'utilitzarà un dataset creat per la Universitat d'Aegean (Grècia) amb el nom d'AWID 2.

S'utilitza aquest dataset perquè és un dataset molt complet i bastant utilitzat en diferents projectes, on hi ha representats una gran diversitat d'atacs Wi-Fi. També consta d'una bona documentació, ja sigui la proporcionada pels propis creadors del dataset [7] o d'altres projectes que l'han utilitzat.

Per a recopilar les dades que es troben al dataset, es va crear i utilitzar un laboratori que simulava la infraestructura d'una petita empresa (Figura 8), també conegut com a "SOHO infrastructure" en anglés. La idea era que els clients connectats a la xarxa de l'empresa anessin fent les seves tasques normals mentre un únic atacant duia a terme diferents tipus d'atacs tant a la xarxa en general com a dispositius concrets.

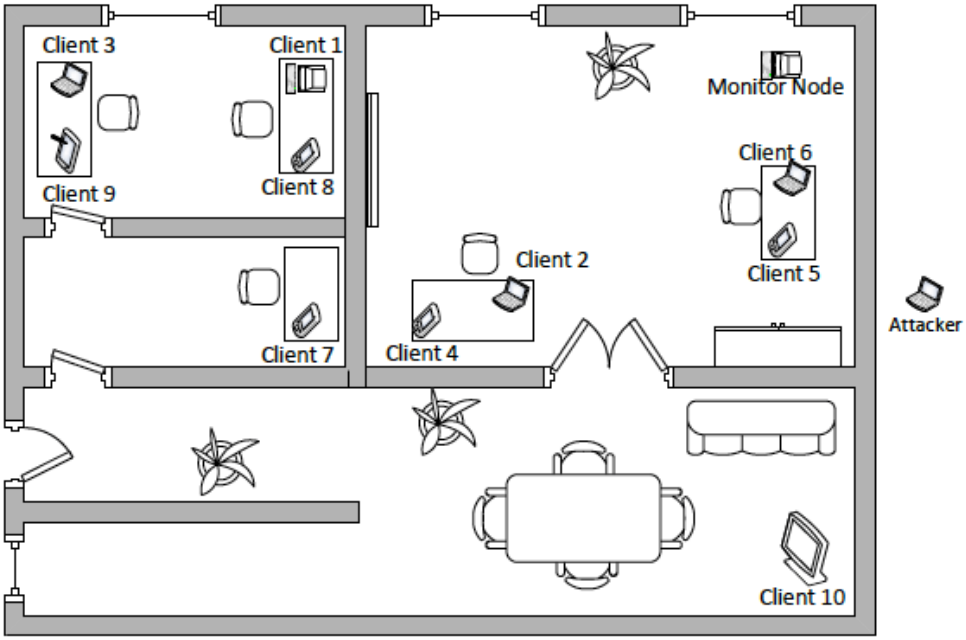


Figura 8: Entorn de creació del dataset

Concretament, aquesta infraestructura constava de:

- Un ordinador de sobretaula.
- Dos ordinadors portàtils.
- Dos smartphones.
- Una tablet.
- Una Smart TV.

Per a capturar el trànsit de la xarxa (ja fos el trànsit normal dels clients o bé el trànsit corresponent a algun dels atacs) es va afegir un node extra a la xarxa que tenia instal·lat el programa Tshark, que permetia generar fitxers en format ".pcap" de les captures que s'anaven fent del trànsit.

Les especificacions tècniques dels dispositius de la xarxa eren les representades a la Figura 9.

Node	Type	Brand	OS	Network Card	CPU
Client1	Desktop	Custom	Ubuntu Linux 12.04 LTS	Netgear WNA3100 N300	Intel Core i7 3.2GHz
Client2	Laptop	Fujitsu-Siemens	Ubuntu Linux 12.04 LTS	Intel 3945ABG	Intel Core Duo T2050 1.6GHz
Client3	Laptop	Acer	Ubuntu Linux 12.04 LTS	Qualcomm Atheros AR9462	Intel Core i5 1.7GHz
Client4	Smartphone	iPhone 3G	iOS 4.2	NA	Samsung 32-bit RISC ARM 620MHz
Client5	Other	iPod Touch	iOS 3.1	NA	Samsung 32-bit RISC ARM 533MHz
Client6	Laptop	Acer Aspire 5750G	Windows 7	Broadcom BCM943227HM4L	Intel Core i5 2.8GHz
Client7	Smartphone	HTC Diamond	Windows Phone 6.1	NA	528 MHz ARM 11
Client8	Smartphone	Samsung Nexus	Android 4.2	NA	dual-core ARM Cortex-A9 1.2 GHz
Client9	Tablet	Samsung Galaxy Tab	Android 2.2	NA	Cortex-A8 1 GHz
Client10	Smart TV	LG 42LM7600S	Linux	NA	NA
Attacker	Laptop	Acer Aspire 5750G	Kali Linux 1.0.6	D-Link DWA-125/Linksys WUSB54GC	Intel Core i5 2.8GHz
Monitor Node	Desktop	Custom	Linux Debian 7.3	Alpha AWUS036H	Core i7 2.4Ghz

Figura 9: Especificacions dels dispositius

Durant les proves, l'ordinador de sobre taula i la SmartTV es van mantenir estàtics, mentre que els smartphones s'anaven movent pel laboratori entrant i sortint de la xarxa repetidament. Per altra banda els ordinadors portàtils solien estar sempre al mateix lloc però de tant en tant es movien de posició.

Pel que fa a la xarxa, tota l'oficina estava connectada a una única xarxa Wi-Fi controlada per un únic AP. Aquesta xarxa rebia diversos atacs de l'atacant (que estava situat a l'exterior de les instal·lacions), el qual utilitzava un ordinador portàtil amb Kali Linux instal·lat i anava canviant d'@MAC de manera habitual.

Un cop obtingudes les captures en format ".pcap", es va realitzar un pas intermedi per a transformar-les a format ".csv", que és el format sobre el qual es treballarà en aquest projecte a l'hora de crear el model.

També es va afegir un camp extra que fa referència a la classe del paquet (pe., si és normal, si pertany a algun atac concret, si pertany a un conjunt d'atacs, etc.).

6. Estudi i decisió de l'atac Wi-Fi

Un dels primers passos del projecte és el de decidir un atac Wi-Fi sobre el qual basar el programa de detecció. L'atac es decidirà d'entre els atacs que es troben representats al dataset AWID2, que (com ja s'ha dit) serà el dataset que s'utilitzarà tant per entrenar com per testejar el model.

Així doncs, s'ha de fer un estudi i un resum de tots els atacs candidats per poder veure quin és el que s'adequa més a les nostres necessitats, també cal tenir en compte que l'atac escollit s'adeqüi a les possibilitats d'aquest projecte (que no sigui molt difícil ni d'entendre ni de realitzar).

6.1. Estudi dels diferents atacs Wi-Fi

Compresos dins el dataset (explicat en detall en els següents apartats) hi ha representats diferents tipus d'atacs a l'arquitectura 802.11 [7].

Aquests atacs es poden agrupar en 4 grans grups:

- **Atacs d'obtenció de la clau**: Són atacs que es centren en obtenir la clau secreta, on la majoria es basen en monitorar diversos paquets. En aquest sentit hi ha diversos atacs:

    - **Atac FMS**: El primer atac registrat d'aquest tipus, permet obtenir la clau compartida WEP aprofitant-se d'una vulnerabilitat de l'algoritme criptogràfic RC4. Es basa en aprofitar les vulnerabilitats d'IV dèbils, on l'atacant assumeix el valor del byte n+1 de la clau d'encriptació quan té coneixement del primer byte del keystream (o flux de claus) i dels n bytes de la clau.

    - **Família d'atacs KoreK**: Són atacs basats en els mateixos principis matemàtics que els atacs FMS però són més eficients. Es basen en aconseguir molta informació dels IV abans de realitzar l'atac com a tal. A vegades s'utilitza aquest atac (i el d'FMS) per a limitar les possibilitats de la clau i aplicar un algoritme de força bruta.

- **Atacs PTW**: Ataca a la versió genèrica de l'algoritme RC4 i intenta obtenir la clau d'una manera molt més eficient que els 2 atacs anteriors. Aquest atac està restringit a paquets ARP, per això s'aplica conjuntament amb altres tècniques com l'anomenada "ARP injection".

- **ARP injection**: No és un atac com a tal, però és una tècnica que (com s'ha vist en el cas anterior) pot complementar alguns atacs. Es basa en manipular la xarxa de manera que es vagin produint nous IV's encara que realment no hi hagi trànsit real, això comportarà que l'atacant pugui capturar alguns d'aquests IV's i els pugui passar al/s seu/s algoritme/s per a seguir amb l'atac de manera offline.

- **Atacs de diccionari**: És una variant dels atacs de força bruta que permet obtenir contrasenyes WPA o WPA2. En una primera instància l'atacant "s'infiltra" en una xarxa per intentar obtenir un handshake en temps real, alternativament també pot atacar a una víctima amb trames de desautenticació per a que aquesta es vegi obligada a fer un handshake a 4 bandes. En el primer cas l'atac és irrastrejable mentre que en el segon si que es podria arribar a detectar (és difícil). En el següent pas, l'atacant utilitza un diccionari per a generar una possible clau per a utilitzar en el 3er missatge del handshake a 4 bandes i provar si funciona. Per a cada clau que provi, n'avalua el resultat i si surt bé podrà obtenir la clau PSK de la xarxa. Cal a dir que aquest atac està restringit només a xarxes que facin servir el mètode PSK, i a sobre el diccionari ha de ser molt gran.

- **Atacs al keystream**[3]: A xarxes de tipus WEP es possible beneficar-se simplement del keystream tot sol, sense la clau compartida. En base a això també hi ha bastants atacs:

[3] Keystream: També anomenat com a flux de claus, fa referència a un flux de caràcters aleatoris o pseudoaleatoris (generats a partir de la clau de xifrat i l'IV) que es combinen amb el flux de text normal per a produir el text xifrat.

- **Atac ChopChop (Figura 10)**: Permet a un atacant obtenir els m últims bytes tant del keystream com del text en clar d'un paquet sense saber-ne la clau. L'atac es basa en que l'algoritme de xifrat CRC-32 s'utilitza malament en WEP i que aquest no ofereix protecció contra respostes a paquets que ja han estat enviats. Com el seu nom indica, l'atac consisteix en tallar "chop" l'últim byte de la part encriptada d'un paquet i intentar deduir el valor del text xifrat d'aquest byte. Després, com que faltarà un byte, la trama tindrà un ICV[4] dolent. En aquest moment, l'atacant fa una XOR amb el paquet truncat amb un valor escollit provant a veure si el resultat porta a una seqüència vàlida per l'ICV específic. Llavors, l'atacant injecta el paquet amb el possible ICV a la xarxa de manera que l'AP hauria de respondre amb un missatge de que no és vàlid (si és el cas) indicant així si l'ha encertat o no.

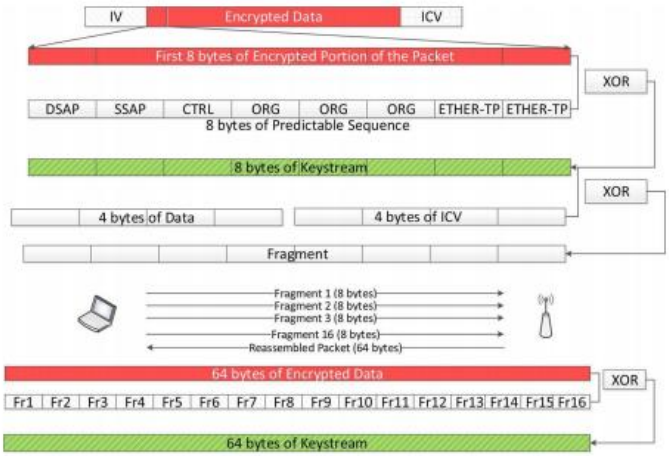


Figura 10 [7]: Esquema de l'atac ChopChop

- **Atacs de fragmentació**: Tenen com a objectiu obtenir una porció significativa del keystream enviant molts menys missatges que l'atac ChopChop. Entre altres coses, permet injectar paquets a la xarxa i donar inici a altres atacs. Aquest atac es basa en que a l'arquitectura 802.11 qualsevol paquet que supera la mida màxima és fragmenta en paquets més petits independents, això s'ajunta amb el fet que els primers 8 bytes de qualsevol trama de dades són fàcils de predir.

[4] ICV: De l'anglès "Integrity Check Value", és una cheeksum de 4 bytes que permet verificar el resultat de la desencriptació.

- **Atac Caffe Latte**: Un atac que es basa en aconseguir la clau WEP sense haver d'estar físicament dins el rang de la xarxa atacada, ja que simplement cal atacar un client aïllat que s'hagi autenticat mínim una vegada a la xarxa objectiu (encara que no hi estigui connectat al moment). És basa en 3 conceptes, el primer és que el client sol tenir una llista de tots els ESSIDs coneguts, el segon és que els clients busquen activament aquestes xarxes quan no estan autenticats (permetent veure la llista de xarxes que tenen guardada), i finalment també es basa en el fet que un client intentarà connectar-se a una xarxa si aquesta té el mateix ESSID que alguna de les que té guardades.

- **Atacs de disponibilitat**: Normalment anomenats DoS (Denial of Service), són atacs que acostumen a tenir com a objectiu un client concret o bé "saturar" els recursos d'una xarxa. En aquest tipus d'atacs cal destacar que per una banda no és un atac permanent i per altra que l'atacant ha d'estar físicament dins el rang de la xarxa atacada mentre duri l'atac.

  - **Atac de desautenticació**: L'atac més simple d'aquest tipus, es basa en el fet que els paquets de desautenticació no estan protegits i es poden suplantar, de manera que quan el client rep un paquet d'aquests abandona la xarxa immediatament sense cap altre acció necessària. L'atac consisteix en que l'atacant monitora el trànsit per a obtenir l'@MAC tant de la víctima com de l'AP per tal de construir un paquet de desautenticació que faci fora la víctima de la xarxa.

  - **Atac de dissociació**: Molt semblant a l'atac de desautenticació però amb un paquet de dissociació, és menys eficient i té una duració més reduïda.

  - **Atac de "broadcast" de desautenticació**: És el mateix que l'atac de des autenticació simple però en comptes d'atacar un client concret a través de la seva @MAC s'ataca a tota la xarxa a través de l'@broadcast (FF:FF...).

- **Atac de "de broadcast" de dissociació**: Com en el cas anterior, és un atac igual que el de dissociació simple però aplicat a tots els usuaris de la xarxa (adreça broadcast) en comptes de un usuari concret. Com en el cas de l'atac de dissociació simple, és menys eficient i dura menys que el de des autenticació.
- **Innundació de bloqueig ACK**: Pot provocar que un AP descarti de manera voluntària tots els paquets d'un client específic aprofitant el mecanisme "Add Block Acknowledgement" (ADDBA) de l'arquitectura 802.11n. Aquest mecanisme es basa en que permet als clients enviar un gran bloc de trames d'un sol cop en comptes d'enviar segments més petits. Per a poder fer això el client envia un missatge ADDBA a l'AP avisant de la intenció d'enviar un sol bloc gran, on aquest missatge conté informació com la mida del bloc que es vol enviar. Després de rebre aquest missatge, l'AP només acceptarà trames que coincideixin amb la seqüència indicada i en descartarà la resta. Basant-se en aquest principi, l'atacant falsificarà una trama ADDBA amb l'@MAC del client i una mida de bloc molt gran, provocant que qualsevol trama que enviï la víctima (que no serà el que l'AP està esperant) sigui descartada (ja que no coincidirà amb la mida esperada). És un atac difícil de detectar ja que ja funciona fent petites injeccions a la xarxa.

- **Inundació de peticions d'autenticació**: L'agressor intenta saturar els recursos de l'AP causant overflow a la taula d'associació del client. Es basa en que el màxim nombre de clients que pot mantenir un AP simultàniament és limitat i depèn d'un valor prèviament codificat o bé de la memòria física de l'AP. Per dur a terme l'atac, l'atacant emularà grans quantitats de clients i enviarà a l'AP trames d'autenticació en nom de cada un d'ells (els "suposats clients"), cosa que provocarà un overflow a l'AP d'entrades falses, el qual no es podrà associar amb els clients legítims.

- **Atacs del fals estalvi d'energia**: A diferència dels altres atacs, aquest es basa en trames de dades nul·les. S'aprofita del mecanisme d'estalvi d'energia fent que l'AP es pensi que un client concret està en mode

"adormit", normalment s'entra en aquest mode quan un client està un temps sense comunicar-se. Per entrar en aquest mode, el client ho notifica a l'AP amb una trama nul·la amb el bit "Power Save" a 1, en aquest mode el client ni enviarà in rebrà informació (ja que la que li hagués d'arribar la guardarà temporalment l'AP). L'atac consisteix en enviar una trama nul·la amb el bit de "Power Save" a 1, l'AP acceptarà el missatge immediatament i començarà a guardar tota la informació que hauria d'anar a la víctima, fent que a aquesta no rebi la informació que espera.

- **Atacs d'inundació CTS**: És un atac que es basa en el mecanisme (opcional) de la parella RTS (Request to send) i CTS (Clear to send). L'atac en si consisteix en enviar la trama RTS per aconseguir accés durant un temps, un cop fet això l'atacant transmet constantment trames CTS cap a ell mateix o cap a altres clients forçant a la resta de clients a aplaçar les seves transmissions de manera continuada.

- **Atac d'inundació RTS**: Com en el cas anterior, aquest atac també s'aprofita del mecanisme RTS/CTS però en aquest cas l'atacant envia un gran nombre de trames falsificades ("spoofed") RTS per tal de monopolitzar la xarxa.

- **Atac d'inundació beacon** : És un tipus d'atac DoS que es pot fer de 2 maneres diferents i que pot provocar "molèsties" a la xarxa o que directament pot provocar la caiguda del servei. La primera versió de l'atac consisteix en enviar un flux constant de trames beacon amb ESSIDs inexistents, comportant que hi hagi un overflow a la llista de xarxes disponibles, provocant problemes per a que un usuari pugui localitzar la xarxa que prefereixi. La segona versió consisteix en que l'atacant enviï una inundació de trames falses ("spoofed") beacon amb un ESSID específic que correspongui a un BSSID inexistent.

- **Atac d'inundació a la "probe request"**: Un atac que té com a objectiu estressar els recursos d'un AP per finalment acabar paralitzant-lo. Aquest atac és possible degut a que un AP està obligat a respondre un missatge de petició amb un missatge de resposta. Així doncs, si l'atacant envia constantment un flux de paquets de petició en un gran volum i durant un espai de temps prolongat provocarà que l'AP es saturi i no pugui respondre a les peticions legítimes.

- **Atac d'inundació a la "probe response"**: És basa en el mateix principi que l'atac anterior però en aquest cas l'atacant transmet una inundació de respostes falses i inexactes als clients.

- **Atacs "Man in the middle**": Són un tipus d'atacs que requereixen més intervenció humana, ja que l'atacant ha d'estar més present en les situacions d'aquest tipus d'atac. D'aquest tipus en destaquen els següents:

  - **Honeypot:** Són xarxes creades i controlades per a administradors amb intencions malicioses. L'objectiu és atraure usuaris ingenus a la xarxa i un cop dins aplicar diferents atacs cap a ells. L'avantatge (per als atacants) de controlar una xarxa és que al no haver-hi encriptació tot el trànsit és visible i permet aplicar de manera senzilla atacs molt específics. Cal a dir que un Honeypot com a tal no és un atac sino un terme global que engloba atacs d'aquest estil.

  - **"Evil twin"**: És un tipus de Honeypot que mostra un ESSID fals per a enganyar a usuaris ingenus per a connectar-se a la seva xarxa en comptes de a la xarxa "bona". Es basa en que pot haver-hi dos o més AP amb el mateix ESSID a la mateixa xarxa, provocant que alguns clients es connectin a un AP (amb l'ESSID igual que el que busquen) que tingui millor qualitat de connexió sense fixar-se en si el BSSID és el bo.

  - **"Rouge access point"**: Són AP sense autorització activats a una xarxa sense el permís de l'administrador.

6.2. Elecció de l'atac

Un cop estudiats i resumits els diferents atacs compresos en el dataset AWID2 cal triar-ne un (que estigui representat en aquest dataset) per a realitzar el treball.

També cal que l'atac escollit pugui ser reproduïble dins el marc d'aquest projecte, amb suficients eines per a realitzar-ne proves reals.

El primer pas és decidir el tipus de l'atac d'entre els 4 grups que s'han vist a l'apartat anterior (Atacs a la clau, atacs al keystream, atacs de denegació de servei i atacs "man in the middle").

D'entrada es descarten tant els atacs a la clau com els atacs al keystream, ja que són atacs bastant més complexos que requereixen coneixements més avançats sobre el funcionament dels mecanismes i algoritmes de xifratge. Al ser tan complexos s'haurien d'estudiar i aprendre més a fons, cosa que comportaria passar més temps del previst en aquesta part. Això podria provocar que s'hagués de dedicar menys temps de l'esperat a altres parts del treball (pe. el desenvolupament i implementació del propi model de detecció d'intrusions), i que perdessin importància.

Per altra banda hi ha els atacs de "man in the middle", que tot i que alguns són més senzills d'entendre i dur a terme no són atacs tant "habituals", ja que requereixen d'una preparació prèvia més gran i s'han de dur a terme "in situ" (almenys la majoria d'ells).

Per tant, descartant els altres tres tipus, s'ha decidit escollir un atac dins el grup dels atacs de denegació de servei, ja que són un tipus d'atac molt freqüent i sobre els quals hi ha molta documentació.

Dins dels atacs de denegació de servei s'ha decidit escollir l'atac de desautenticació, sobre el qual es basarà el sistema de detecció d'intrusions. S'ha escollit aquest atac degut a que és dels més habituals d'aquest tipus, es basa en un tipus de trames de les més habituals (trames de desautenticació), i consta de molta documentació relacionada (tant per com detectar-lo com per com dur-lo a terme). També és un tipus d'atac que està bastant present al dataset AWID2 que farem servir (ja es veurà més endavant) i és relativament "fàcil" de dur a terme en cas de poder fer la part de simular-ho en un escenari real.

6.3. Ampliació sobre l'atac de desautenticació

Un cop seleccionat l'atac de desautenticació com a base per al nostre sistema de detecció és convenient explicar i conèixer una mica millor com funciona aquest atac. Cal dir que tot i que no faria falta saber com funciona l'atac de desautenticació per a poder-lo detectar (això ho aprendrà automàticament el model), és important entendre com funciona per a si més endavant l'hem de "simular" en un escenari real.

L'atac de desautenticació es basa en utilitzar (com el seu nom indica) les trames de gestió del tipus de desautenticació. Aquestes trames les envia l'AP cap a un client, que quan la rebi es desconnectarà automàticament de la xarxa sense poder dur a terme cap altra acció addicional, tampoc podrà fer cas omís del que li diu la trama. El motiu pel qual existeixen aquestes trames és que l'AP tingui el poder de fer fora un client de la xarxa quan ho cregui necessari (pe. quan el client porta massa estona inactiu, quan considera que el client està connectat a la xarxa sense autorització, etc.) [13].

Una altra cosa a tenir en compte de les trames de desautenticació és que no estan protegides (no estan xifrades), això vol dir que es pot accedir a la informació que contenen, com l'@origen o l'@destí. Això a priori pot no semblar un problema, però comporta que siguin un tipus de trames molt fàcil de suplantar.

La combinació del funcionament d'aquestes trames i el fet que siguin fàcils de suplantar és el que provoca quer sigui un dels atacs més utilitzats, i dels més "senzills" de realitzar.

Per a realitzar-lo (Figura 11), l'atacant haurà d'esbrinar tant l'AP al que la víctima està connectat com l'@MAC d'aquesta. A partir d'aquí enviarà una trama de desautenticació a la víctima suplantant a l'AP provocant que s'hagi de desconnectar automàticament de la xarxa i hagi de tornar a dur a terme el procés d'autenticació per a tornar-se a connectar. Si aquest atac es repeteix suficients vegades, pot provocar que la víctima no només es desconnecti de la xarxa, sinó que no pugui tornar a establir-hi connexió de manera indefinida (mentre duri l'atac).

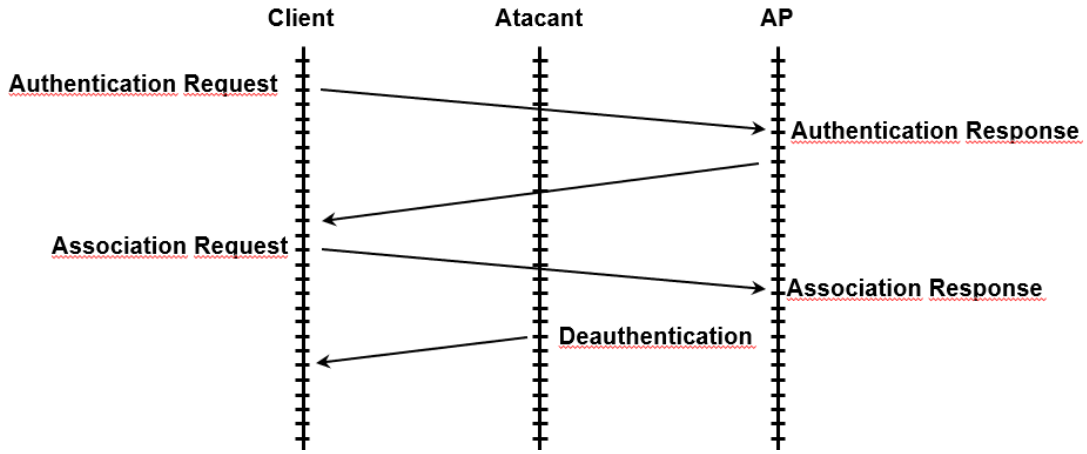


Figura 11: Exemple d'atac de desautenticació

Una altre manera de realitzar l'atac (Figura 12) és enviant a l'AP una trama de petició de desautenticació suplantant a la víctima de manera que, l'AP enviï una trama de desautenticació totalment legítima a la víctima, provocant el mateix resultat que en el cas anterior, que la víctima es desconnecti de la xarxa automàticament.

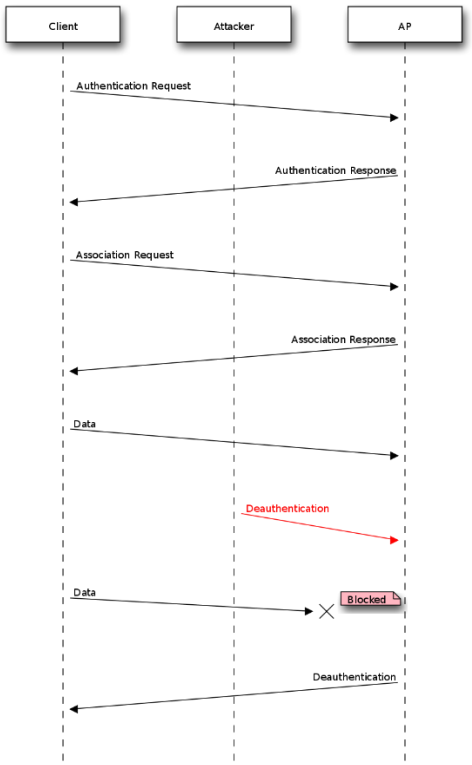


Figura 12: Un altre exemple d'atac de desautenticació

6.4. Què fer en cas de patir l'atac

Ara que es coneix el funcionament de l'atac pot sorgir la pregunta de si hi ha alguna manera d'evitar-lo. La resposta és que no (o és molt difícil), no es pot evitar completament l'atac, ja que aprofita una debilitat intrínseca del protocol 802.11, és una debilitat que no es pot eliminar, ja que s'eliminaria una funcionalitat d'aquest protocol.

Per evitar l'atac s'hauria de bloquejar l'atacant directament, però aquest en molts casos és difícil de detectar (i encara és més difícil bloquejar-lo). El que sí que es pot fer és actuar i mitigar-ne els efectes, on el grau de mitigació depèn de les mesures que estigui disposat a prendre cadascú (pe., es poden reforçar les parets de l'empresa per tal que no entrin connexions Wi-Fi de l'exterior, es poden intentar bloquejar manualment les @ que puguin ser de l'atacant, etc.).

Tot i no ser completament mitigable, és important detectar-lo de seguida per tal d'actuar en conseqüència, no només per a intentar reduir-ne els efectes sinó per a intentar buscar alternatives a la possible pèrdua de connexió (pe., habilitar un altre AP, passar a connexió per cable mentre duri l'atac, etc.). Aquí és on entra un bon sistema de detecció.

Finalment, cal dir que tot i que (com s'ha dit abans) l'atac és molt difícil d'evitar, el nou protocol WPA3 (que molt pocs AP implementen) i algunes versions de l'WPA2 implementen una millora anomenada "Protected Management Frames" (PMF) [10], que (com el seu nom suggereix) protegeix algunes trames de gestió (com les de desautenticació) de manera que no es pugui accedir a la seva informació i no es puguin suplantar.

Hi ha una ampliació de la informació sobre les PMF als annexos.

6.5. Altres maneres de detectar l'atac

Es poden detectar els atacs Wi-Fi d'altres maneres que no sigui mitjançant Deep Learning, de fet utilitzar Deep Learning és una iniciativa que ha estat prenent força recentment.

Existeixen molts programes i llibreries per a poder capturar i analitzar el trànsit en temps real. Aquests anàlisis passen una sèrie de filtres predefinits, els quals són resultat de molt d'estudi del funcionament i la metodologia d'aquests atacs.

Aquests filtres combinats amb bones eines de monitoratge permeten controlar el trànsit d'una xarxa de manera molt eficient, essent així capaços de detectar amb molta precisió diferents atacs.

Un exemple de com detectar aquests atacs [23] és escanejar el trànsit mitjançant Wireshark, on s'ha de posar la targeta Wi-Fi del dispositiu en mode monitor mitjançant el programa airmon-ng i el programa airodump-ng. S'han de realitzar els següents passos:

- Posar la targeta Wi-Fi de l'ordinador en mode monitor i escanejant el canal adequat.
- Capturar el trànsit de la interfície desitjada mitjançant Wireshark.
- Afegir un filtre de color pels tipus de paquets que interessin. Per exemple, a la Figura 13 es veu com els paquets de dades es marquen en verd, els de desautenticació en taronja i els de dissociació en groc.

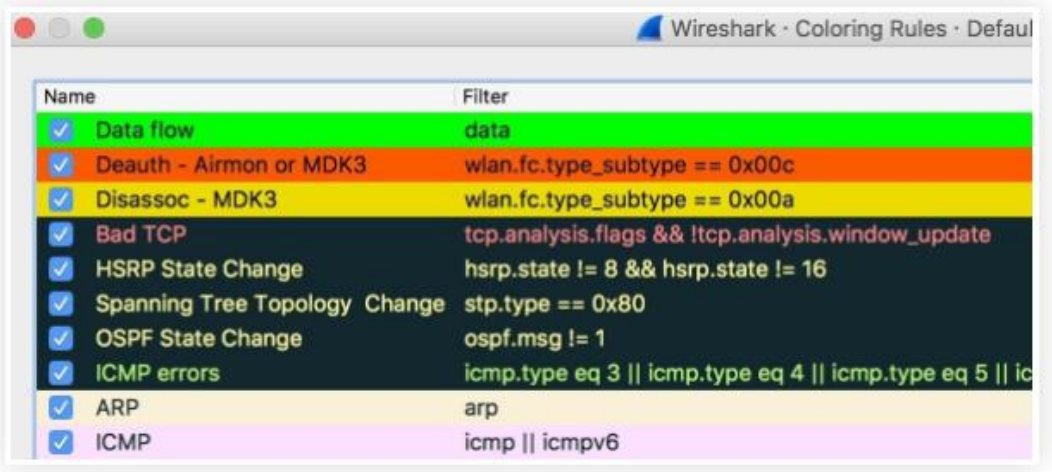


Figura 13: Exemple de filtres de colors al Wireshark [23]

- Si es produís un atac, al Wireshark sortiria una captura com la de la Figura 14, on es poden veure moltes trames de desautenticació seguides (en aquest cas marcades en groc i no en taronja). Això és un indicatiu d'alerta que (tot i no ser molt sofisticat) permetria veure que alguna cosa anormal està passant.

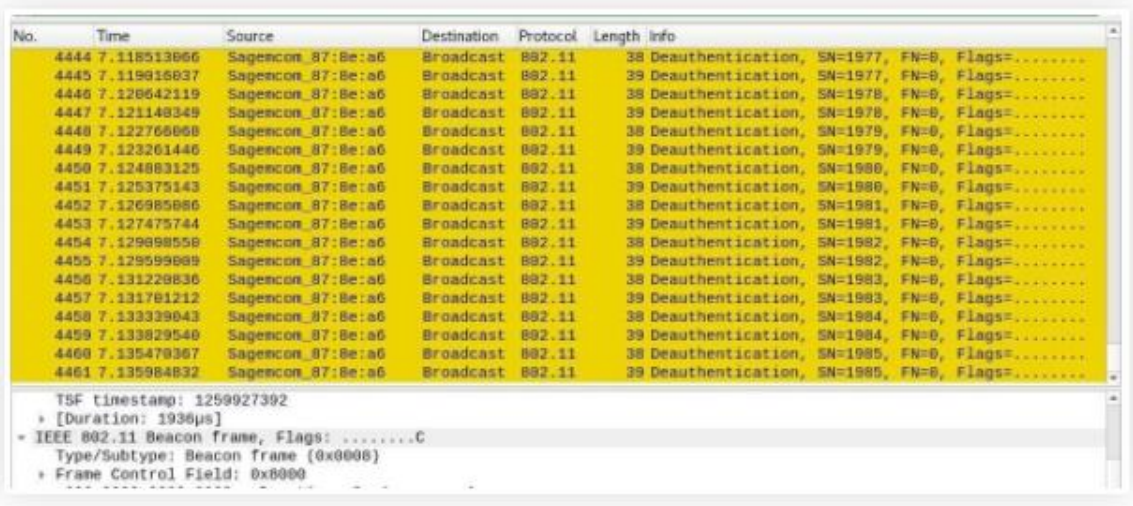


Figura 14: Exemple d'atac de desautenticació amb filtres [23]

Aquest exemple d'atac és una cosa molt senzilla, però que representa molt bé com funcionen altres mètodes de detecció que no estiguin basats en Deep Learning. Els sistemes més utilitzats automatitzen tots els passos anteriors i disposen de filtres molt més sofisticats, però la idea és la mateixa.

Tot i això, el fet de fer-ho utilitzant Deep Learning ofereix molta més versatilitat perquè és el mateix programa el que aprèn en funció de l'exemple i que és capaç d'extreure els filtres, a sobre va millorant-se i aprenent de manera constant, cosa que pot permetre que s'adapti als atacs més nous.

7. Anàlisi, disseny i implementació

7.1. Requisits del sistema

7.1.1. Requisits de Hardware

Com s'ha explicat a l'apartat d'estudi de viabilitat, a nivell de hardware el sistema no ha de constar d'uns requisits específics més enllà dels següents:

- **Suficient espai de disc per a guardar-hi els datasets**: Tot i ser fitxers de text, els datasets poden arribar a ser força "pesats", de manera que si es tenen molts i molts datasets amb moltes dades s'ha d'arribar a disposar de suficient espai per a guardar-ho tot. En aquest projecte (com veurem més endavant) tampoc cal gaire espai, ja que no s'utilitzaran gaires datasets i tampoc seran massa pesats.
  El servidor disposa d'un espai d'emmagatzematge d'1 TB HDD i 500 GB SSD a compartir entre tots els usuaris. Així que és més que suficient per a guardar tot el codi i els datasets (que en aquest projecte no ocupen gaire més de 100MB).

- **Suficient capacitat de computació per a entrenar un model que consti d'una xarxa neuronal**: Aquest requisit no demana d'uns valors concrets ja que depenent del model els requisits seran més o menys exigents, però en termes generals de com més memòria RAM es disposi millor i també tenir una targeta gràfica dedicada pot ser un bon "plus" (o depenent de quins models d'IA pot ser un requisit indispensable).
  El servidor consta de 16 GB de RAM, de les quals 15 estan disponibles per a repartir entre tots els usuaris. També consta d'una targeta gràfica dedicada "nVidia GeForce GTX 1070 (8 GB GDDR5)".
  Són especificacions molt bones per a dur a terme el projecte perfectament.

Cal dir que en projectes més grans amb aquests requisits no seria suficient, ja que s'arriben a utilitzar clústers sencers amb centenars de GB de RAM i una alta quantitat de computació per a entrenar models molt més complexos i amb moltes més dades.

7.1.2. Requisits de Software

El programa ha de permetre/fer els següents punts:

- Entrar les dades del trànsit Wi-Fi tant d'entrenament com de test en el mateix format que les del dataset AWID2 (s'explicarà en els pròxims apartats).
- Llegir i tractar les dades entrades.
- Definir les característiques del model de Deep Learning.
- Entrenar el model a través de les dades d'entrenament entrades.
- Poder visualitzar diferents valors del rendiment del model (p.e., el learning rate, gràfiques que mostrin si hi ha overfitting, etc.).
- Poder testejar i visualitzar els resultats i les prediccions del model sobre les dades de test.

Més enllà dels requisits de les coses que "ha de fer" el programa, hi ha altres requisits que ha de tenir el model:

- Un cop entrenat, el model ha de ser capaç de generalitzar el que ha après per a dades diferents de les d'entrenament obtingudes d'altres escenaris (però en el mateix format que les d'entrenament).
- El programa ha de dur a terme tant l'entrenament com les prediccions en un temps "correcte". No hauria d'estar dies processant.

El programa ha de ser suficient "genèric" dins del que sigui possible. Encara que estigui basat en un atac concret, no s'han d'haver de canviar gaires coses per tal de poder-lo aplicar a un altre atac.

7.2. Disseny del programa

A l'hora de dissenyar el programa es diferenciaran els següents blocs funcionals (Figura 15):

- **Input**: Es processen i es llegeixen les dades del dataset.
- **Entrenament**: Es defineix el model i s'utilitzen les dades de la fase anterior per a entrenar-lo.

- **Output**: S'obtenen els resultats de l'entrenament i es realitzen una sèrie de proves i comprovacions (pe., obtenir-ne l'eficiència, comprovar si hi ha overfitting, buscar el percentatge de falsos positius...).

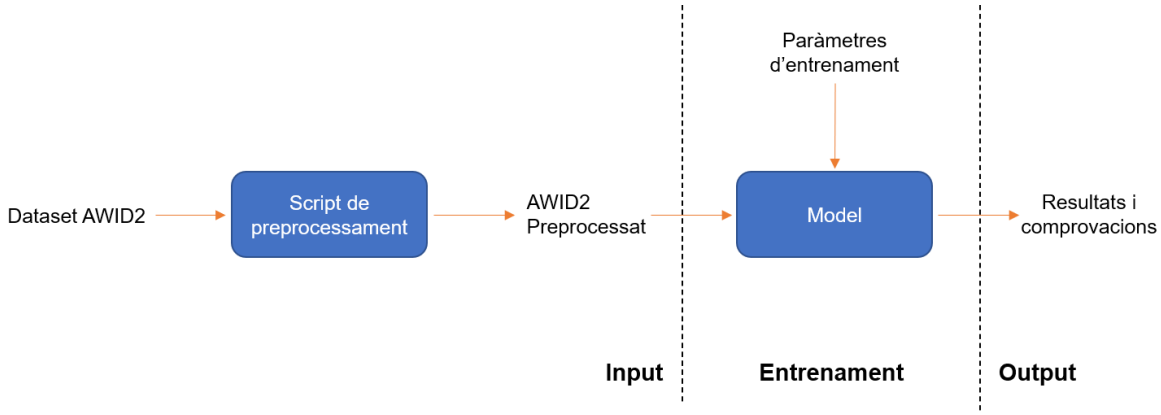


Figura 15: Blocs de l'aplicació

Per tal que aquests tres blocs funcionin es duran a terme els següents passos en ordre seqüencial (Figura 16):

1. **Preprocessament del dataset**: Es realitzaran una sèrie d'accions sobre el dataset per tal d'eliminar dades innecessàries i fer-lo més "llegible" per al programa. Aquestes accions es realitzaran en un programa diferent al de la resta de blocs.
2. **Lectura i tractament de les dades**: Dins del programa, s'han de definir diversos paràmetres (pe., d'on s'obtenen les dades, quina és la variable a predir, quines són les variables contínues, etc.).
3. **Definició i entrenament del model**: Cal indicar els paràmetres del model i representar les mètriques necessàries per a veure quins són els valors adequats de cada un d'ells. Després definir el nombre d'iteracions (epochs) i entrenar el model.
4. **Comprovacions sobre el model**: Comprovar diversos paràmetres del model (pe., comprovar la precisió, comprovar si hi ha overfitting, etc.).

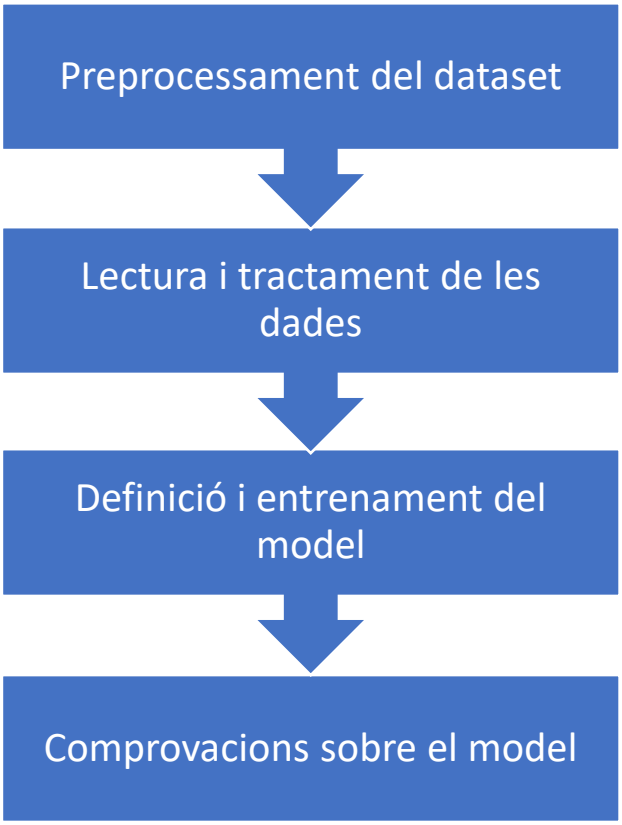


Figura 16: Blocs funcionals del programa

7.3. Estructura i representació del dataset

El primer pas a realitzar és el d'analitzar el dataset, entendre com s'ha creat aquest dataset, quines dades conté i com estan estructurades, també s'ha d'analitzar com s'hi troben aquestes dades representades [12].

7.3.1. Estructura del dataset

Els fitxers en format ".csv" explicats a l'apartat anterior es troben agrupats dins una sèrie de subcarpetes dins una gran carpeta comprimida (Figura 17).

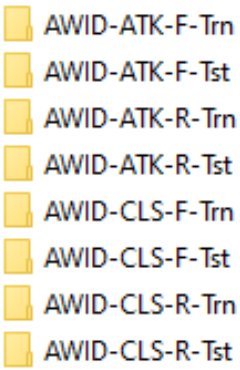


Figura 17:Estructura del dataset

Les subcarpetes estan estructurades de la següent manera:

- Datasets d'atacs ("ATK"):
    - Datasets complets ("F"):
        - Dataset d'entrenament ("Trn")
        - Dataset de test ("Tst")
    - Datasets reduïts ("R")
        - Dataset d'entrenament ("Trn")
        - Dataset de test ("Tst")
- Datasets de classificació ("CLS"):
    - Datasets complets ("F"):
        - Dataset d'entrenament ("Trn")
        - Dataset de test ("Tst")
    - Datasets reduïts ("R")
        - Dataset d'entrenament ("Trn")
        - Dataset de test ("Tst")

Els datasets del tipus "ATK" tenen representats els diferents atacs explicats anteriorment de manera individual. Es a dir, tots els paquets de les captures realitzades es poden classificar dins el rang de tots els atacs explicats a l'apartat 6 (pe., caffe latte, evil twin, deauthentication, fragmentation, etc.) o com a paquets de trànsit "normal".

Per altra banda, els datasets de tipus "CLS" classifiquen els atacs de l'apartat 6 en tres grans grups, de manera que els paquets capturats es poden representar dins un d'aquests tres grups o en un quart grup que representarà als paquets que són normals. Així doncs, un paquet podrà ser classificat com a "flooding", "impersonation", "injection" o "normal".

Aquest tipus de classificació és la que s'anomena classificació per metodologia "By Methodology" (Figura 18) i a part de les tres classes d'atac representades al dataset ("flooding", "impersonation" i "injection") hi ha una quarta classe anomenada "passive" però que no té representació al dataset, ja que són atacs que no deixen rastre i no es poden detectar.

Attacks	By Purpose				By Target		By Methodology			
	Key Cracking	Keystream	DoS	M-i-M	Network	Client	Passive	Injection	Flooding	Impersonation
FMS	✓				✓		✓			
Korek	✓				✓		✓			
PTW	✓				✓		✓			
ARP Injection	✓				✓			✓		
Dictionary	✓				✓		✓			
Chop-Chop		✓			✓			✓		
Fragmentation		✓			✓			✓		
Caffe Latte		✓			✓					✓
Hirte		✓			✓					✓
Deauthentication			✓			✓			✓	
Disassociation			✓			✓			✓	
Disassociation			✓			✓			✓	
Deauthentication broadcast			✓			✓			✓	
Disassociation broadcast			✓			✓			✓	
Block Acknowledge			✓			✓			✓	
Authentication Request			✓			✓			✓	
Fake Power Saving			✓			✓			✓	
CTS			✓			✓			✓	
RTS			✓			✓			✓	
Beacon			✓			✓			✓	
Probe Request			✓			✓			✓	
Probe Response			✓			✓			✓	
Honeypot				✓		✓				✓
Evil Twin				✓		✓				✓
Rogue AP				✓		✓				

Figura 18: Classificació dels atacs per metodologia [7]

Com que per aquest projecte es volen detectar únicament els atacs de desautenticació, interessa que al dataset hi hagi els atacs d'aquest tipus representats de manera explícita i no agrupats dins una classe que també inclogui altres atacs (com passa amb el dataset de tipus "CLS"). Per tant es descarten els datasets de tipus "CLS" i s'utilitzaran els del tipus "ATK" que si que tenen els atacs representats de manera individual.

Dins dels datasets "ATK" hi ha la versió "F" i la versió "R". La "F" fa referència a "Full", i significa que és el dataset complet amb tots els fitxers csv resultants de la totalitat de les captures realitzades.

Per altra banda la "R" fa referència a "Reduced", que és una versió més reduïda del dataset però també prou significativa com per utilitzar-la en un model.

S'ha decidit utilitzar la versió reduïda "R" del dataset, ja que conté dades de sobra per a la tasca que es vol realitzar i no requereix tant d'espai d'emmagatzematge ni tant de temps per a processar-lo.

Dins de cada un dels tipus de datasets hi ha un dataset que s'utilitzarà per a l'entrenament ("Trn") i un per al test ("Tst"). Això permetrà que per provar el funcionament del model se li puguin entrar dades que no hagi "vist", tot i que no passaria res si es validés el model mitjançant el dataset d'entrenament, ja que a

l'hora d'entrenar es separaria un percentatge del dataset d'entrenament per a validació el qual no s'utilitzaria per a entrenar.

En resum, els datasets utilitzats en el model seran l'"AWID-ATK-R-Trn" i l'"AWID-ATK-R-Tst" (Figura 19).

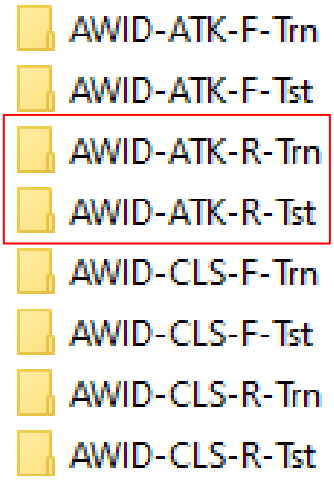


Figura 19: Datasets seleccionats per al model

7.3.2. Representació de les dades al dataset

És molt important entendre en quin format estan representades les dades al dataset per a saber com llegir-les i tractar-les. També cal mirar si s'han de processar les dades abans de poder-les llegir.

En aquest cas, les diferents captures estan representades en format tabular. És a dir, tots els camps de les captures estan representats en una taula on cada fila representa un paquet i cada columna un atribut del paquet (Figura 20).

Dins el format tabular, les dades es guarden en fitxers en format ".csv" (coma separated values), on les files de la taula es representen com a files, però les diferents columnes se separen amb comes per a poder-les diferenciar (Figura 20).

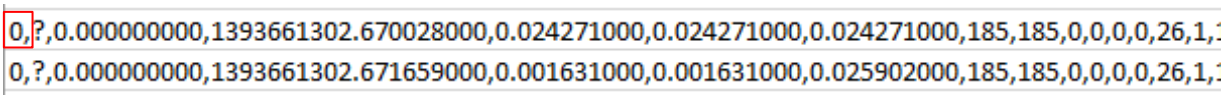


Figura 20: Línies del dataset d'entrenament I (atributs del principi)

A part dels diferents atributs Wi-Fi dels paquets, hi ha una columna extra anomenada "class" (Figura 21) que representa la classe del paquet (pe., si és un paquet normal, si és resultat d'un atac de desautenticació, etc.).

```
0x0007,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,deauthentication
?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,normal
```

Figura 21: Línies del dataset d'entrenament II (atributs del final on es pot veure l'atribut "class")

Com és normal, no tots els paquets tenen un valor definit a tots els diferents camps, ja que depenent del tipus de paquet contindran diferent informació i atributs.

Per representar això, cada paquet (cada fila) tindrà valor als camps que realment tingui valor i tindrà un '?' als altres camps (Figura 22).

```
0x0007,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,deauthentication
?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,normal
```

Figura 22: Línies del dataset III (Atributs sense valor)

Cada fila consta dels 155 atributs diferents que poden tenir els paquets Wi-Fi (pe., frame_time_delta, frame_len, wlan.ra, etc.), entre els quals hi consta l'atribut extra que fa referència la classe del paquet anomenat "class", atribut qual s'afegeix a posteriori de les captures [7].

Cal remarcar també, que no hi ha cap fila que indiqui els noms de les diferents columnes per a saber quin és el nom de cada atribut.

7.4. Implementació de l'script de preprocessament

7.4.1. Preprocessament de les dades

Tal com es pot veure a l'apartat anterior, el dataset conté les dades capturades molt "crues" (Figura 23) ja que les columnes no estan etiquetades. També s'ha de tenir en compte que molts d'aquests atributs no són necessaris per a detectar un atac de desautenticació o directament podrien provocar un entrenament erroni del model.

```
0,?,0.000000000,1393661302.670028000,0.024271000,0.024271000,0.024271000,185,185,0,0,0,0,26,1,
0,?,0.000000000,1393661302.671659000,0.001631000,0.001631000,0.025902000,185,185,0,0,0,0,26,1,
```

Figura 23: Exemple de files del dataset

Així doncs, el primer que s'haurà de fer abans de res és netejar i processar les dades de manera que serveixin per a entrenar el model per a l'atac de desautenticació. Això ho farem (com s'ha dit) amb un script a part.

Per fer aquest script s'agafarà com a referència un projecte realitzat per un usuari de github anomenat Brandon Marlowe (amb username "Bee-Mar") [14]. En el seu projecte també crea un programa de detecció d'intrusions, però amb altres tècniques d'IA que no són de Deep Learning.

Cal dir que del seu projecte només s'agafarà un script que utilitza per a netejar el dataset que també es pot aprofitar per a aquest projecte.

Els passos a realitzar a l'script per a obtenir el dataset processat que llegirà el model són els següents:

1. **Decidir quins atributs mantenir**:
   Cal indicar quins atributs són necessaris per a detectar un atac de desautenticació dins dels 155 i descartar-ne la resta. Per a decidir quins atributs són necessaris ens basem en els mateixos que agafa en Brandon, ja que ell obté bons resultats fent-ho d'aquesta manera i són atributs bastant lògics (més endavant es veurà que representen). Aquests atributs són els corresponents a les columnes "2, 5, 45, 62, 64, 65, 68, 71, 74, 75, 88, 91, 92, 105, 106, 110, 116, 120, 154" (Figura 24).

```
desired_cols = [2, 5, 45, 62, 64, 65, 68, 71, 74,
                75, 88, 91, 92, 105, 106, 110, 116, 120, 154]
```

Figura 24: Selecció de les columnes desitjades

2. **Afegir una fila inicial amb els noms dels atributs de les columnes**:
   El fet de no poder relacionar cada columna del dataset amb el seu nom en perjudica el tractament. Per tant, afegir una fila al principi del dataset que contingui els noms de tots els atributs permetrà poder identificar millor els atributs que hem mantingut.

El primer que es fa és mirar quins són els noms d'aquestes columnes (Figura 25).

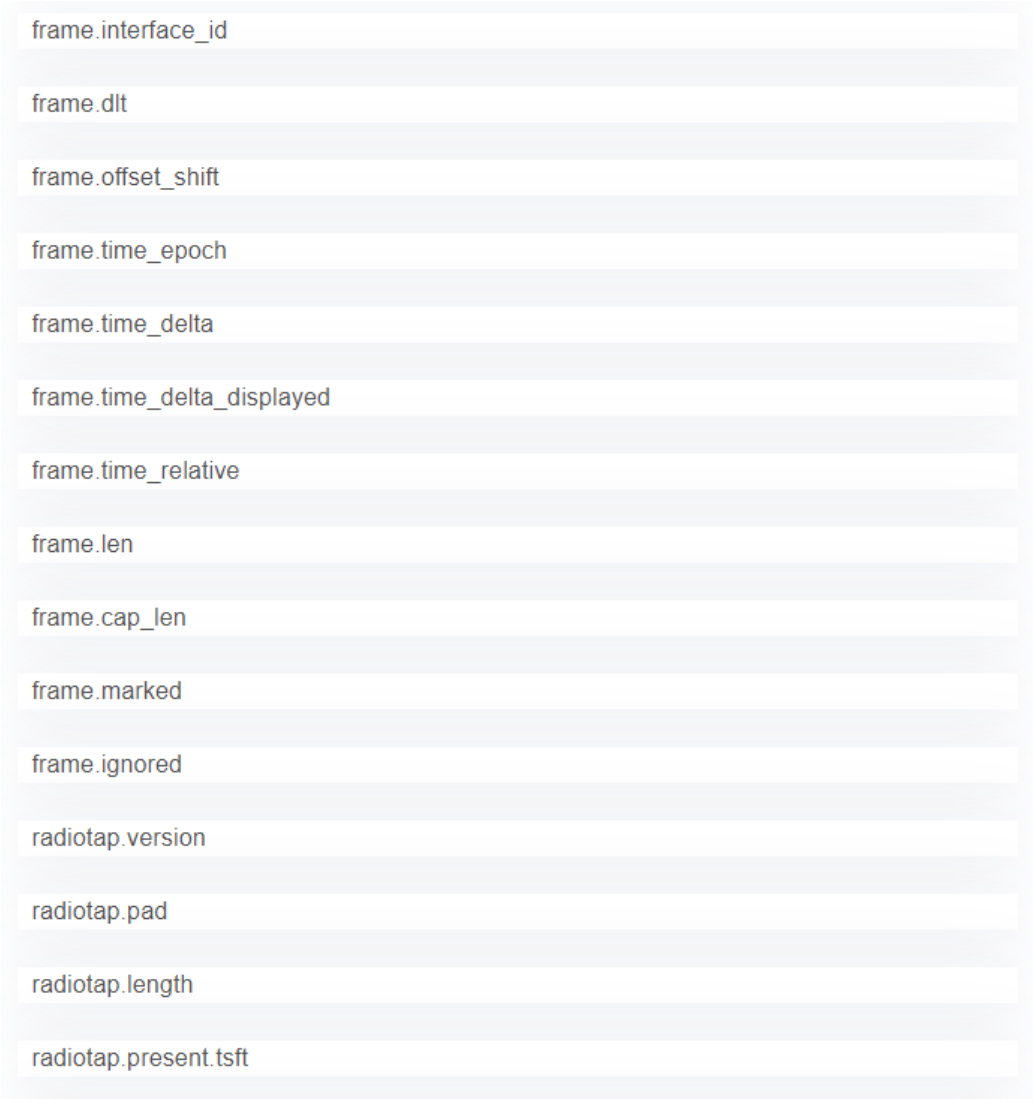


Figura 25: Exemple dels noms dels atributs Wi-Fi [7]

Després es passen tots aquests noms a un fitxer anomenat "colnames.txt". Partint d'aquest fitxer, l'script afegirà una fila al principi del dataset amb el nom corresponent de la columna que toqui (Figura 26).

```
with open(Path(resource_dir, 'col_names.txt')) as cols:
    for line_num, col_name in enumerate(cols):
        if line_num in desired_cols:
            col_names.append(col_name.rstrip())
```

Figura 26: Codi d'afegir els noms de les clumnes

3. **Substituir tots els '?' per 'NaN' i eliminar totes les columnes amb més d'un 60% d'NaN**:

El model creat amb *fastai* no interpretarà els '?' com a valors inexistents i que ha d'ignorar, per tant cal indicar-li que són valors no definits identificant-los com a NaN (Not a Number) (Figura 27). També cal eliminar les columnes amb més d'un 60% dels valors NaN, ja que no són suficientment útils com per introduir-les en el model (Figura 28).

```
data.replace('?', np.nan, inplace=True)
```

Figura 27: Codi per substituir els '?' per ' NaN'

```
prev_num_cols = len(data.columns)
data.dropna(axis='columns', thresh=len(data.index) * 0.40, inplace=True)
print("Removed " + str(prev_num_cols - len(data.columns)) +
      " columns with all NaN values.")
```

Figura 28: Codi per a eliminar les columnes amb més d'un 60% d'NaN

4. **Eliminar les columnes que tinguin un valor constant**:
   Una columna amb un valor constant no és representativa, no es pot estudiar en quines circumstàncies canvia de valor perquè al tenir valor constant no canvia mai de valor (Figura 29).

```
cols_to_drop = []

for col in data:
    if not data[col].nunique() > 1:
        cols_to_drop.append(col)

data.drop(columns=cols_to_drop, inplace=True)
```

Figura 29: Codi on s'eliminen les columnes amb valors constants

5. **Eliminar les files (o paquets) que contenen almenys un atribut amb valor "NaN"**:
   No interessen els paquets amb valors no definits, ja que pot portar problemes a l'hora de tractar-los i buscar-ne un patró (Figura 30).

```
old_num_rows = data.shape[0]
data.dropna(inplace=True)
```

Figura 30: Codi on s'eliminen totes les columnes amb almenys un atribut amb valor "NaN"

6. **De la columna "class", substituir els elements que tinguin un valor diferent de "deauthentication" per "no deauthentication"**:
   Com que en aquest projecte només detectarem atacs de desautenticació, tenir els altres tipus d'atac en el dataset no interessa. Per evitar això agrupem tots els paquets que no siguin de desautenticació (pe., els paquets normals, els paquets arp, els paquets evil twin, etc.) sota un mateix nom, "no deauthentication". D'aquesta manera l'atribut "class" només podrà estar entre dos valors, ['deauthentication', 'no deauthentication'] (Figura 31).

```
data["class"] = data["class"].replace(['amok', 'arp', 'beacon',
'beacon', 'cafe_latte', 'evil_twin', 'fragmentation', 'normal',
'normal', 'probe_response'], 'no deauthentication')
```

Figura 31: Substitució dels valors de la columna "class" que no siguin del tipus "deauthentication"

7. **Exportar el fitxer resultant**:
   Un cop realitzat el filtratge i tots els canvis, cal exportar el fitxer resultant en un fitxer amb extensió ".csv" (Figura 32) per tal que el programa de detecció el pugui llegir.

```
data.to_csv(
    Path(resource_dir, 'preproc_dataset-un-sol-atac.csv'),
    index=False,
    sep=',')
```

Figura 32: Codi per passar el resultat a csv

7.4.2. El dataset preprocessat

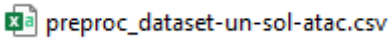


Figura 33: Exemple del fitxer resultant

Un cop executat el programa anterior s'obté el fitxer "preproc_dataset-un-sol-atac.csv" (Figura 33), aquest fitxer serà el que s'utilitzarà per entrenar les dades. Per obtenir el fitxer de test executem el mateix programa però canviant tant el nom de la ruta d'origen com el de la ruta destí.

Respecte al dataset d'entrenament (ja preprocessat), aquest consta finalment de 10 atributs d'entre els 155 inicials (Figura 34).

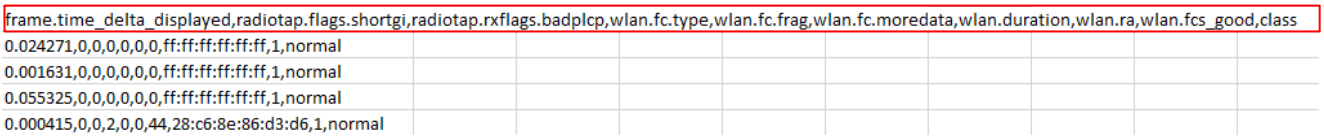


Figura 34: Exemple de les primeres files del dataset d'entrenament

Aquest atributs són:

- **frame.time_delta_displayed [15]**: Indica el temps de diferència del paquet actual respecte al paquet anterior.
- **radiotap.flags.shortgi [16]**: Fa referència a les propietats de les trames enviades i rebudes. Concretament fa referència a una propietat anomenada "Guard Interval"[5] (gi).
- **radiotap.rxflags.badplcp [16]**: Fa referència als anomenats "RX flags" i pot pendre 3 valors en funció de les propietats de les trames rebudes.
- **wlan.fc.type [17]**: Fa referència al tipus de trama, amb un 0 representa les trames de gestió, amb un 1 les de control i amb un 2 les trames de dades.
- **wlan.fc.frag [18]**: Indica si venen més fragments associats al paquet actual.
- **wlan.fc.moredata [18]**: Indica si l'AP conté guardades dades que s'haurien d'enviar al client.
- **wlan.duration [18]**: Indica la duració en microsegons que fa referència a la trama actual.
- **wlan.ra [18]**: Fa referència a la "Reciever address", que indica l'@MAC del següent node del camí per arribar al destí (pe., l'AP).

[5] Guard interval: és un petit interval de temps que es deixa a l'inici i final d'una transmissió per evitar possibles interferències

- **wlan.fcs_good [18]**: Indica el resultat de realitzar la "Frame Check Sequence"[6] (FCS).
- **class**: Com ja s'ha vist, fa referència a la classe que a la que pot pertànyer cada paquet que, després del preprocessament només pot ser del tipus "deauthentication" o del tipus "no deauthentication".

Més enllà de les columnes eliminades, també s'han eliminat moltes files ja que la majoría tenien valors NaN.

En total queden 1.793.602 files en comparació a les 1.795.575 files inicials. S'han eliminat un total de 1.973 files.

7.5. Implementació del sistema de detecció d'intrusions

7.5.1. Lectura i tractament de les dades

El primer pas en la creació del sistema és definir les llibreries, llegir i tractar les dades d'entrada. Per fer-ho s'han de realitzar una sèrie de passos:

- **Importació de llibreries**:
  Dins *fastai* no utilitzarem totes les seves opcions, només farem servir les que estan relacionades amb les dades en format tabular. Per tal de fer servir totes les funcionalitats que es veuran més endavant s'han d'importar els paquets que es poden veure a la Figura 35.

```
from fastai.tabular.all import *
from fastai import *
```

Figura 35: Paquets que s'importen

- **Llegir el fitxer ".csv" amb les dades**:
  Creem una variable sota el nom de "train" a la que li assignem el resultat de llegir el fitxer "preproc_dataset-un-sol-atac.csv" (Figura 36). Després també mostrem les primeres files del dataset per comprovar que les dades s'hagin entrat en el format correcte (Figura 37).

[6] Frame Check Sequence: S'utilitza per a que el receptor de les dades sàpiga que aquestes no es troben corruptes.

```
In [2]: train = pd.read_csv('preproc_dataset-un-sol-atac.csv')
        train.head()
```

Figura 36: Lectura del dataset d'entrenament

Out[2]:

	frame.time_delta_displayed	radiotap.flags.shortgi	radiotap.rxflags.badplcp	wlan.fc.type	wlan.fc.frag	wlan.fc.moredata	wlan.duration	wlan.ra	wlan.fc
0	0.024271	0	0	0	0	0	0	ff:ff:ff:ff:ff:ff	
1	0.001631	0	0	0	0	0	0	ff:ff:ff:ff:ff:ff	
2	0.055325	0	0	0	0	0	0	ff:ff:ff:ff:ff:ff	
3	0.000415	0	0	2	0	0	44	28:c6:8e:86:d3:d6	
4	0.000005	0	0	1	0	0	0	00:25:bc:ed:07:cf	

Figura 37: Representació de les primeres files de les dades llegides

- **Llegir el fitxer amb les dades de test**:
  Tal com es fa amb el dataset d'entrenament, lo ideal hagués sigut llegir i utilitzar el dataset de test que proporcionava l'AWID2 per a validar el model, però a l'hora d'utilitzar-lo sortia un error, que s'explicarà més en detall a l'apartat de problemes trobats. A causa d'aquest error, el dataset de test no es pot fer servir i s'ha optat per reservar una part del dataset d'entrenament per a validació. No hi ha cap problema en fer servir el dataset d'entrenament, ja que és suficientment gran com per poder-li treure una part sense perdre informació o dificultar-ne l'entrenament.
- **Indicar quina és la variable dependent, quines són les variables categòriques i quines són les variables contínues:**
  Cal indicar al programa quina és la variable que volem predir (anomenada variable dependent) per tal que centri l'aprenentatge sobre aquesta columna. També s'ha d'indicar quines variables (columnes) tenen valors continus i quines valors categòrics. Això és necessari perquè les variables categòriques es tracten de manera diferent a les contínues, ja que estan expressades de

manera diferent (les categòriques es mouen sempre dins un rang finit de categories mentre que les contínues poden prendre qualsevol valor).

Figura 38: Definició de la variable dependent, les variables categòriques i les variables continues

- **Definir els processos a realitzar sobre les dades**:
  Per a poder entrenar el model amb les dades llegides, fa falta realitzar alguns processaments més (Figura 39). En primer lloc és important realitzar un procés de categorització (Categorify), aquest procés transforma les variables categòriques (definides al punt anterior) en variables contínues. També interessa emplenar les caselles on hi hagi algun valor buit (tot i que en principi no faria falta, ja que ja s'ha controlat a l'script de preprocessament), aquest procés s'anomena "FillMissing". Finalment cal normalitzar tots els valors per a poder-los tractar de manera uniforme en un procés anomenat "Normalize".

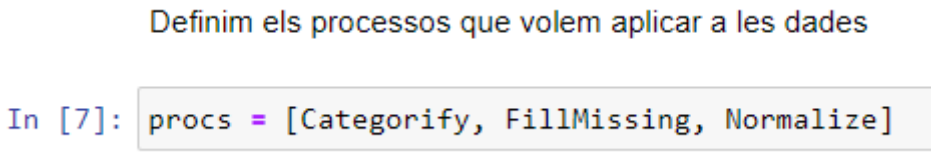


Figura 39: Definició dels processos a realitzar sobre les dades

- **Partir les dades d'entrada en una part d'entrenament i una part de test**:
  Degut als problemes del dataset de test explicats anteriorment usarem el dataset de train tant per a entrenar el model com per a validar-lo. Per fer això reservarem una part del dataset que no s'utilitzarà per entrenar, sinó que quedarà guardada per a validar el model amb dades que no hagi vist mai. Les proporcions seran un 80% del dataset per entrenar i un 20% per validar (són les proporcions més usades). Per indicar això implementem la comanda de la Figura 40.

Definim les particions

```
In [8]: splits = RandomSplitter(valid_pct=0.2)(range_of(train))
```

Figura 40: Definició de les particions del dataset

- **Passar pel TabularPandas:**

  Un cop hem definit tots els paràmetres i els processos a realitzar ho hem de passar per el TabularPandas. Aquest procés generarà un objecte que després podrem convertir a un format que el nostre model podrà llegir. Per tal que pugui generar aquest objecte, s'ha d'indicar el lloc d'on es treuen les dades (la variable train), quines són les variables contínues, categòriques i la variable dependent (que hem assignat en els passos previs), i quines particions farem al dataset. La comanda que genera aquest objecte (a la variable "to") és la que es pot veure a la Figura 41.

Construïm el TabularPandas

```
In [9]: to = TabularPandas(train, procs = procs, cat_names = var_cat, cont_names = var_cont, y_names = var_dependent, splits = splits)
```

Figura 41: Creació de l'objecte mitjançant el TabularPandas

- **Convertir les dades en un dataloader:**

  Convertirem l'objecte creat amb el TabularPandas (to) en un dataloader (Figura 42), de manera que el model pugui llegir les dades (que ara si que estan totalment tractades). A l'hora de transformar l'objecte en un dataloader podem passar-hi diversos paràmetres que volem que no agafin el valor per defecte, en aquest cas només modificarem el paràmetre anomenat "bs" que fa referència a la "batch size"[7]. Per aquest model creiem que una bs de 4096 és suficient com per a poder-lo entrenar a una bona velocitat sense risc d'overfitting.

```
In [10]: dls = to.dataloaders(bs=4096)
```

Figura 42: Creació del dataloader

[7] Batch Size: És la quantitat de dades que es llegiran simultàniament a cada iteració. S'ha d'anar en compte amb aquesta variable ja que un valor molt petit pot introduir soroll al model mentre que si és molt gran pot comportar overfitting.

7.5.2. Definició i entrenament del model

Un cop ja tenim les dades entrades i en el format adequat anem a una de les parts més importants de totes: la de crear, definir i entrenar el model.

A partir d'aquí les opcions de personalització que es poden aplicar per crear el model són moltes, però en aquest projecte tampoc es configuraran manualment gaires coses, ja que és un tema que pot arribar a ser molt complex. A part, la llibreria permet deixar moltes opcions per defecte, ja que està feta per treballar a molt alt nivell. Així doncs, analitzarem els paràmetres més importants i deixarem els altres en valor per defecte.

El primer pas és crear un "learner" [21]. Aquí és on teòricament es defineixen tots els paràmetres relacionats amb la xarxa neuronal, des d'on s'obtenen les dades d'entrada fins a altres atributs com el nombre de capes i neurones per capes.

En aquest projecte, del learner no modificarem gaire res, només les capes neuronals (Figura 43), on constarà de 2 hidden layers (una amb 200 neurones i l'altre amb 100 neurones) a part de la input i la output layer. Aquests valors són els que s'utilitzen per defecte, de fet si no els especifiquéssim s'agafarien aquests igual, ja que els assignaria *fastai* directament per defecte. Bàsicament els especifiquem per fer-ho més visual i deixar clar en tot moment l'estructura de la xarxa. També hem d'indicar quin dataloader farà servir i, en el nostre cas, també indiquem que a part de les mètriques que es mostren per defecte durant l'entrenament també mostri la precisió.

```
In [12]: learn = tabular_learner(dls, [200, 100], metrics = accuracy)
```

Figura 43: Definició del learner

Un cop definit el learner és interessant veure'n el learning rate ja que ens pot permetre interpretar si el model és bo o si cal ajustar-ne algun paràmetre.

Per mostrar el learning rate apliquem la comanda de la Figura 44.

```
In [13]: learn.lr_find()

Out[13]: SuggestedLRs(lr_min=0.09120108485221863, lr_steep=0.0020892962347716093)
```

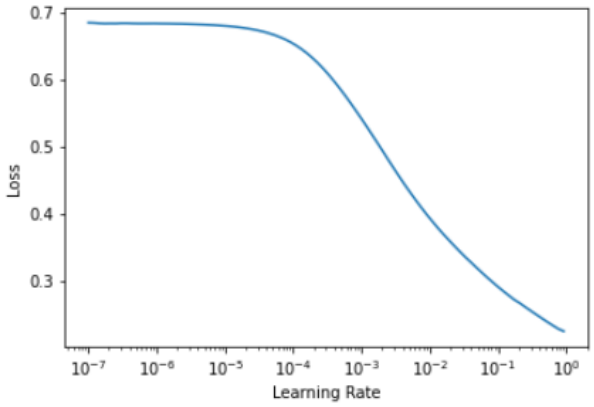


Figura 44: Obtenció del learning rate al nostre programa

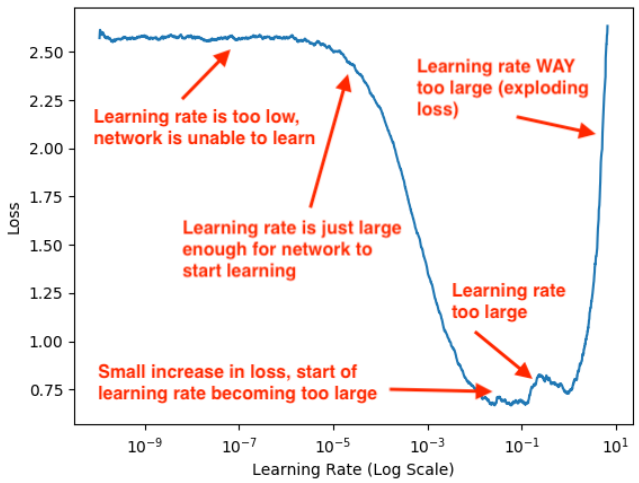


Figura 45: Exemple explicació del learning rate

Per definir la gràfica de la Figura 44 el programa comença entrenant un model a un learning rate baix i el va augmentant progressivament. Si ens fixem a la gràfica en qüestió, a mesura que augmenta el learning rate disminueixen les pèrdues. Aquest és el comportament correcte, ja que si ens fixem en la Figura 45 (que és un exemple del comportament normal d'aquest tipus de gràfiques) el model no és capaç d'aprendre a learning rates baixes (tal com surt a la Figura 44). Un cas que no es representa a la Figura 44 i que si que sol ser el comportament habitual (Figura 45) és quan el learning rate és molt elevat, on el model tampoc és capaç d'aprendre (en el nostre cas no s'arriba a aquest punt). En resum, a mesura que passa el temps, com més aprèn el model menys pèrdues té.

Un cop hem vist que el model està ben definit procedim a entrenar-lo, per fer-ho utilitzarem un mètode anomenat fit_one_cycle. Dels paràmetres que se li poden passar a aquest mètode només modificarem el nombre d'epochs (iteracions).

Per decidir el nombre d'iteracions s'ha de seguir un procés d'encert i error. Es van incrementant el nombre d'iteracions de manera progressiva fins que s'obtinguin uns resultats d'entrenament esperats, però tampoc es pot augmentar indefinidament, ja que a un cert punt hi podrà haver overfitting.

En aquest projecte s'ha decidit entrenar el model amb 100 epochs. Per obtenir aquest valor es va partir d'un epoch i es va anar augmentant de 10 en 10 fins que van sortir uns resultats acceptables. Sempre es va anar comprovant que no hi hagués overfitting (més endavant es veurà com es comprova si hi ha overfittinfg).

Així doncs, entrenem el model amb la comanda mencionada (Figura 46) i indicant per paràmetre que es facin 100 epochs (els altres paràmetres els deixem per defecte).

```
In [14]: learn.fit_one_cycle(100)
```

Figura 46: Comanda per entrenar el model

De la comanda anterior s'obté la sortida de la Figura 47, on es poden veure una llista d'iteracions i el valors de diferents mètriques, com les pèrdues en l'entrenament, les pèrdues en la validació, la precisió del model i el temps que ha tardat a fer cada iteració.

epoch	train_loss	valid_loss	accuracy	time
0	0.056081	0.045254	0.993942	00:10
1	0.025140	0.024430	0.993942	00:10
2	0.021280	0.022203	0.993942	00:10
3	0.020446	0.021453	0.993942	00:10
4	0.020624	0.021152	0.993942	00:10

Figura 47: Sortida de l'entrenament del model

Els valors de les pèrdues tant d'entrenament com de validació ens serviran per a poder comprovar si hi ha overfitting. Per altra banda, el valor de l'"accuracy" serà un indicador de lo precís que és el model amb dades noves, en aquest cas veiem que ronda el 99% de precisió. Cal dir que aquest valor no és del tot representatiu, ja que engloba la precisió tant al detectar les trames del tipus "no desauthentication" com les de "desauthentication" quan lo millor seria separar els dos valors. Això no vol dir que no haguem de fer cas a aquest valor, ja que és un bon indicador de que el model (almenys de manera global) és molt precís.

Finalment, cal veure l'anomenat "nivell de generalització del model", que indica si aquest està ben entrenat, si hi ha overfitting o si hi ha underfitting. Això és molt important de detectar, ja que un model mal entrenat no podrà realitzar la seva tasca més enllà de les dades d'entrenament i comportarà que (com s'ha vist anteriorment) s'hagin de modificar els valors d'alguns dels paràmetres, com el nombre d'epochs.

Una bona manera de veure el nivell de generalització és mitjançant la representació gràfica de les pèrdues del set d'entrenament i del de validació. Aquestes pèrdues són les que es poden veure a la Figura 47 sota els noms de "train_loss" i "valid_loss" durant totes les iteracions.

Un cop es té la gràfica hi pot haver tres casos:

- **La pèrdua d'entrenament és més gran que la pèrdua de validació**: En aquest cas estem parlant d'overfitting ja que el model està tan entrenat que en comptes d'aprendre es podria dir que ha "memoritzat" les dades.
- **La pèrdua de validació és més petita que la pèrdua de validació**: En aquest cas es tracta d'underfitting degut a que el model no s'ha entrenat suficient com per haver aprés sobre les dades.
- **La pèrdua d'entrenament és semblant a la pèrdua de validació**: En aquest cas l'entrenament és correcte.

Així doncs mitjançant la comanda de la Figura 48 veiem la representació de les pèrdues. Com es pot veure a la mateixa figura, la gràfica que representa les pèrdues d'entrenament és molt semblant a la que representa les pèrdues de validació. Això significa que el model està ben entrenat en aquest sentit.

```
In [15]: learn.recorder.plot_loss()
```

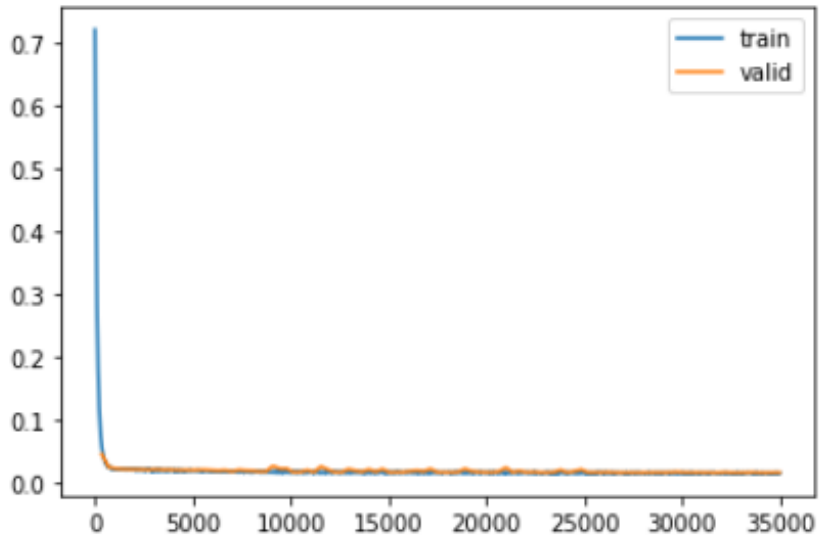


Figura 48: Generalització del model en funció de les pèrdues

8. Proves i resultats

8.1. Validació del model

Un cop tenim implementat i entrenat el model, cal comprovar que fa la seva funció realitzant una sèrie de proves més enllà de les mètriques vistes a l'apartat anterior.

Per fer això es generen prediccions sobre un dataset de test i se n'extreuen diverses mètriques i valors que ajudaran a entendre com de bo és el model o si té algun comportament anormal.

8.1.1. Generació dels resultats

Per a generar els resultats primer cal obtenir un dataset sobre el qual realitzar aquestes prediccions. Per tant, ha de ser un dataset en el mateix format que els que s'han entrat per a entrenar el model.

En aquest cas, com que el dataset de test no funciona (com s'ha explicat en apartats anteriors), s'ha fet servir una copia del dataset d'entrenament. Això no hauria de portar problemes, ja que tal i com s'ha vist a l'apartat anterior, el model està ben entrenat i no ha "memoritzat" les dades, de manera que encara que les dades ja les hagi vist per entrenar-se, no les "recordarà".

Així doncs definim una variable anomenada "test" que contindrà una còpia del dataset d'entrenament. Després, tal i com hem fet amb el dataset d'entrenament, definim un dataloader que contindrà el dataset en un format adequat per a dur a terme les prediccions i analitzar-les. Al dataloader hi definim la mateixa batch size que en el cas de l'entrenament, 4096.

Això ho fem amb les comandes de la Figura 49.

```
In [61]: test = train.copy()

In [62]: dl = learn.dls.test_dl(test, bs=4096)
```

Figura 49: Definició del dataset de proves i del seu dataloader

Un cop creat el dataloader amb les dades de test, ja s'hi poden realitzar les prediccions. Per obtenir-les utilitzem la comanda "learn.get_preds()", la qual

agafa el dataloader que li passis per paràmetre, el llegeix, ignora la columna que es vol predir i genera un objecte amb els valors que ell ha predit de cada línia.

Cal dir que degut al procés de categorització "Categorify" que s'ha definit a l'apartat 9.2.1, els valors de la variable "class" han passat de ser ["deauth-entication, "no deauthenticacion"] a ser representats amb els valors numèrics [0, 1] respectivament.

Mitjançant la comanda de la Figura 50 obtenim les prediccions a la variable "preds".

```
In [112]: preds, _ = learn.get_preds(dl=dl)

In [113]: preds
Out[113]: tensor([[5.3046e-04, 9.9947e-01],
                  [1.2106e-02, 9.8789e-01],
                  [6.2117e-04, 9.9938e-01],
                  ...,
                  [1.1030e-06, 1.0000e+00],
                  [4.2215e-13, 1.0000e+00],
                  [5.5543e-04, 9.9944e-01]])
```

Figura 50: Generació de les prediccions

Com es pot veure a la Figura 50, tot i que ja tenim les prediccions, no estan representades en un format "fàcil" de tractar ja que, entre altres coses contenen informació que no interessa per als anàlisis que es volen realitzar.

Per obtenir les dades que ens interessen, que són les prediccions com a tal, utilitzarem les funcionalitats de la llibreria numpy tal i com es mostra a la Figura 51. El resultat serà una array d'enters entre 0 i 1 fent referència al valor de la variable "class" on 0 és deauthentication i 1 és no deauthentication.

```
In [22]: final_preds = preds.numpy()

In [23]: final_preds = np.argmax(final_preds, axis=1)

In [24]: final_preds
Out[24]: array([1, 1, 1, ..., 1, 1, 1])
```

Figura 51: Obtenció de les prediccions

8.1.2. Anàlisi dels resultats

Un cop tenim les prediccions en un format adequat procedim a aplicar-hi una sèrie d'algoritmes per obtenir-ne diverses mètriques que ens puguin indicar com de bo (o dolent) és el sistema.

Generem un tros de codi que recorre l'array de prediccions i comprova si l'element actual que està mirant està predit com a "deauthentication". De ser així, va a la fila equivalent a la que està mirant del dataset d'entrenament i comprova si allà la classe també pren el valor de "deauthentication". De ser així incrementa el valor d'un comptador anomenat "com_correctes_deauth", si és del tipus "no deauthentication" (significant que ha classificat malament el paquet) incrementa el comptador anomenat "comptador_incorrectes_deauth". També hi ha un altre comptador anomenat "comptador_deauth" que compta el nombre de trames que el programa ha classificat com a "deauthentication" (independentment de si les ha classificat bé o malament).

Com es pot veure a la Figura 52, el programa ha classificat un total de 3.056 trames com a "deauthentication", d'aquestes trames més del 70% les ha classificat correctament mentre que el 30% restant eren trames del tipus "no deauthentication" que ha classificat com a "deauthentication".

```
In [24]: comptador_deauth = 0
         comptador_correctes_deauth = 0
         comptador_incorrectes_deauth = 0
         fila = 0
         for x in final_preds:
             if x == 0:
                 real = train.iloc[fila][9]
                 if real == "deauthentication":
                     comptador_correctes_deauth += 1
                 else:
                     comptador_incorrectes_deauth += 1
                 comptador_deauth += 1
             fila += 1
         print("Total: ", comptador_deauth, "Precisió: ", (comptador_correctes_deauth/comptador_deauth)*100)

         Total:  3056 Precisió:  70.6151832460733
```

Figura 52: Obtenció dels paràmetres per a generar les mètriques I

Que tingui un 70% de precisió en les trames que classifica com a "deauthentication" està molt bé, significa que quan classifiqui alguna trama com a part de l'atac aquesta tindrà moltes probabilitats (70%) de realment estar ben detectada. En altre paraules, no hi ha gaires falsos positius.

Un cop obtingudes les dades en el cas de les trames predites com a "deauthentication", realitzem el mateix processament amb les que prediu com a "no deauthentication".

```
In [25]: comptador_no_deauth = 0
         comptador_correctes_no_deauth = 0
         comptador_incorrectes_no_deauth = 0
         fila = 0
         for x in final_preds:
             if x == 1:
                 real = train.iloc[fila][9]
                 if real == "no deauthentication":
                     comptador_correctes_no_deauth += 1
                 else:
                     comptador_incorrectes_no_deauth += 1
                 comptador_no_deauth += 1
             fila += 1
         print("Total: ", comptador_no_deauth, "Precisió: ", (comptador_correctes_no_deauth/comptador_no_deauth)*100)

         Total:  1790546 Precisió:  99.53706858131542
```

Figura 53: Obtenció dels paràmetres per a generar les mètriques II

En aquest cas (Figura 53) podem veure com classifica 1.790.546 trames dins el tipus "no deauthentication", d'aquestes un 99,5 % estan ben classificades ja que realment no son de desautenticació mentre que el 0,5% restant estan mal classificades.

Aquests percentatge és molt bo, significa que quan classifica una trama com a negatiu (o no desautenticació) ho fa de manera molt precisa i en molts pocs casos s'equivocarà.

Abans de realitzar un anàlisi més complet, calculem dos paràmetres més. Recorrem el dataset de train per comptar el nombre de trames de cada tipus que hi ha realment. A la figura 54 veiem que hi ha 10.447 trames del tipus "deauthentication" i 1.783.155 trames el tipus "no deauthentication".

```
In [26]: real_deauthentication = train.isin(['deauthentication']).sum(axis=0)

In [27]: real_no_deauthentication = train.isin(['no deauthentication']).sum(axis=0)

In [28]: real_deauthentication[9]
Out[28]: 10447

In [29]: real_no_deauthentication[9]
Out[29]: 1783155
```

Figura 54: Obtenció del nombre de trames de cada tipus dins el dataset d'entrenament

Finalment amb les dades obtingudes generem un petit tros de codi que mostri totes les dades simultàniament (incloses el nombre de trames de cada tipus que classifica de manera errònia).

```
In [30]: print("Positius: ", comptador_correctes_deauth, "Realment positius: ", real_deauthentication[9])
         print("Negatius: ", comptador_correctes_no_deauth, "Realment negatius: ", real_no_deauthentication[9])
         print("Falsos positius: ", comptador_incorrectes_deauth)
         print("Falsos negatius: ", comptador_incorrectes_no_deauth)

Positius:  2158 Realment positius:  10447
Negatius:  1782257 Realment negatius:  1783155
Falsos positius:  898
Falsos negatius:  8289
```

Figura 55: Obtenció de més mètriques

A la Figura 55 podem veure que de 10.447 paquets provinents d'un atac de desautenticació en classifica correctament 2.158 mentre que de 1.783.155 paquets provinents de trànsit que no és de desautenticació en classifica correctament 1.782.257.

Això es tradueix en que s'han obtingut 898 falsos positius i 8.289 falsos negatius.

Per la part de falsos positius molt bé, ja que pràcticament no falla al classificar paquets que no venen d'atac com a tal.

Però en relació als falsos negatius no està tant bé, ja que classifica aproximadament un 80% dels paquets provinents d'un atac com a paquets normals, encara que després dels paquets que classifiqui com a atac tingui un bon percentatge d'encert.

A sobre el punt on falla més és amb els falsos negatius, que en termes de seguretat informàtica és més preocupant que els falsos positius. Un fals negatiu significa que un atac és classificat com a no atac (passa desapercebut) i això pot comportar problemes molt més greus que si un paquet que no és maliciós es classifica com a part d'un atac.

Una comparació per fer més entenedor el problema que representen tants falsos negatius seria amb el cas de la covid. És molt pitjor que un pacient amb covid es faci un test que surti negatiu (seguirà fent vida normal i podrà contagiar el virus) que no pas que un pacient sense covid surti positiu.

8.2. Test del model en el cas d'un atac real

Un cop realitzades les diferents proves i anàlisis del model directament, s'ha intentat dur a terme un atac de desautenticació real per passar-lo al sistema de detecció dissenyat.

La idea era realitzar l'atac en un entorn controlat al laboratori, capturar el trànsit per obtenir les captures tant del trànsit normal com de l'atac, passar les captures al format ".csv" amb tots els atributs per tal de poder-les processar i passar les dades preprocessades al sistema de detecció d'intrusions.

Malauradament no s'ha pogut completar aquest últim objectiu. Tot i que si que s'ha aconseguit realitzar l'atac i obtenir-ne les captures, no s'han pogut transformar a format ".csv" conservant tots els atributs originals dels paquets Wi-Fi.

A internet hi ha molta poca documentació al respecte i la referència de la bibliografia dels creadors de l'AWID2 [7] està desactualitzada. Per aquesta raó, a causa del poc marge de temps que quedava, s'ha optat per deixar aquesta part com a treball futur.

Per a realitzar l'atac de desautenticació seguirem els passos d'un tutorial d'internet que utilitza el programa Aircrack [22].

Per no realitzar l'atac a la xarxa Wi-Fi del grup de recerca que utilitza tothom, fem servir un AP diferent que no s'estava fent servir. Com a víctima s'utilitza el meu mòbil, i l'objectiu és aconseguir que aquest es desconnecti de la xarxa Wi-Fi de l'AP.

En relació al software, s'instal·la el programa anomenat anteriorment (Aircrack) a un ordinador portàtil amb Linux. L'ordinador serà el d'en Miquel Farreras, estudiant de doctorat del grup de recerca que ha ajudat a la realització d'aquest projecte. S'utilitza el seu portàtil, ja que ja hi tenia instal·lat el Linux.

Un cop realitzades les preparacions prèvies (tant de hardware com de software) realitzem els següents passos per a dur a terme l'atac:

- Al terminal de Linux escrivim la comanda "iwconfig" per veure informació de la targeta Wi-Fi de l'ordinador (Figura 56).

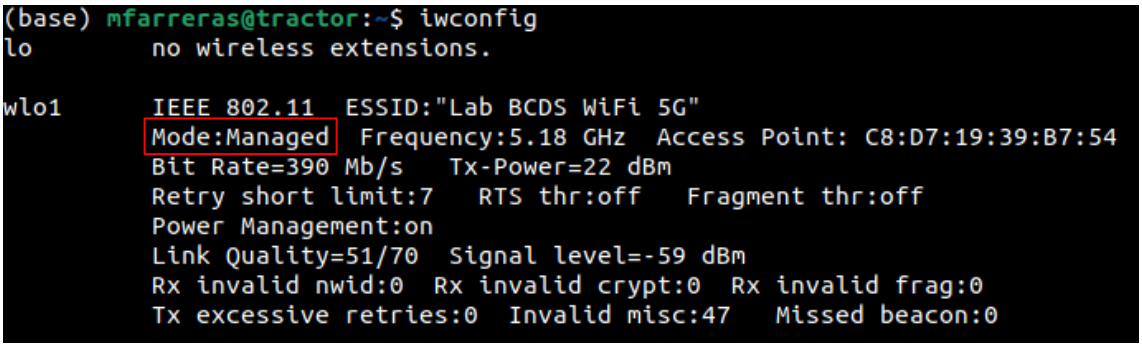


Figura 56: Informació de la targeta Wi-Fi de l'ordinador

- De la informació de la Figura 56 ens interessa el mode de la targeta. Per defecte està en mode "Managed" però per a realitzar l'atac volem que estigui en mode "Monitor".
- Per canviar el mode de "Managed" a "Monitor" s'ha d'executar el programa airmon-ng que venia dins el paquet de l'Aircrack.
- Dins el programa airmon-ng s'ha d'executar la comanda "airmon-ng check kill" (Figura 57), que "matarà" tots els processos que podrien interferir en el correcte funcionament de l'atac.

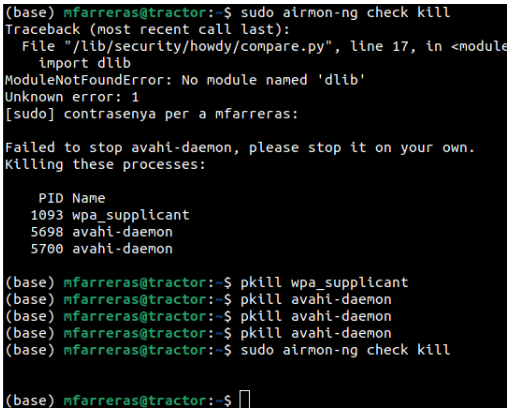


Figura 57: Comanda per matar tots els processos relacionats amb l'atac

- Després iniciem el programa per tal que es canviï la interfície (Figura 58) i tornem a introduïr la comanda "iwconfig" per a comprovar si s'ha canviat a mode "Monitor".

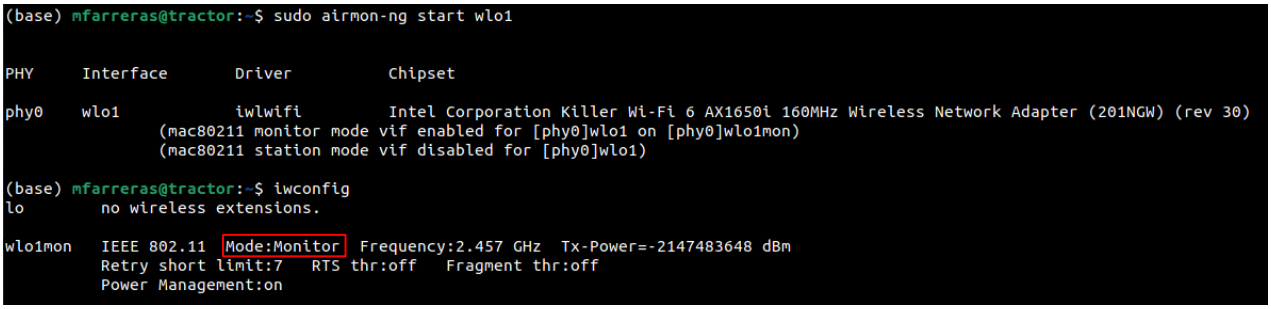


Figura 58: Canvi de mode de l'interfície Wi-Fi

- Ara volem veure tots els AP dels voltants per identificar les dades de l'AP on es troba la "víctima". Per fer això escrivim la comanda "airmon-ng start wlo1" (Figura 59).

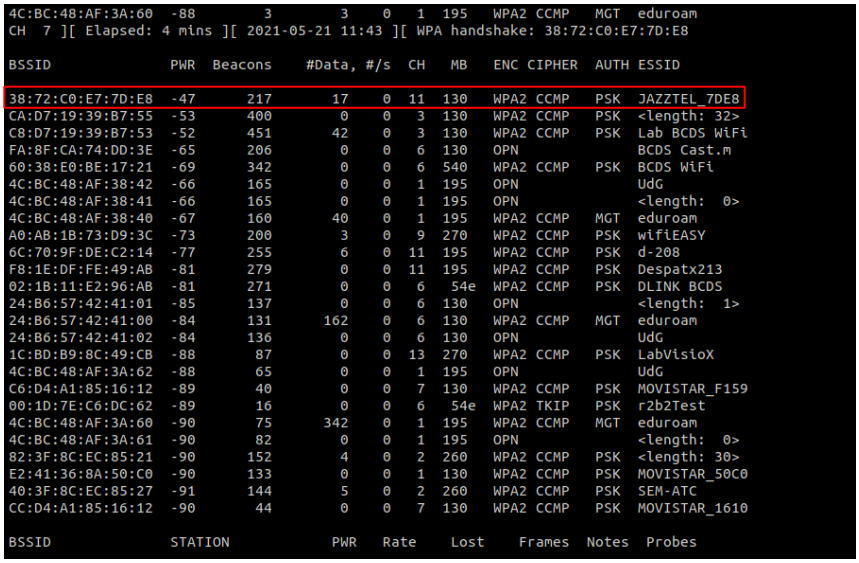


Figura 59: Llista d'informació de la xarxa Wi-Fi

- De la Figura 59 hem de localitzar l'AP que volem atacar i n'anotem l'@MAC i el canal al qual transmet. Podem veure que l'@MAC és "38:72:C0:E7:E8" i el canal és l'11.
- Seguidament volem veure els clients connectats a l'AP (que ja tenim identificat). Per fer això, posem la comanda de la Figura 60 i mostrarà tots els dispositius connectats (Figura 61).

```
mfarreras@tractor:~$ sudo airodump-ng --bssid 38:72:C0:E7:7D:E8 --channel 11 wlo1mon
```

Figura 60: Comanda per veure tots els clients d'un AP

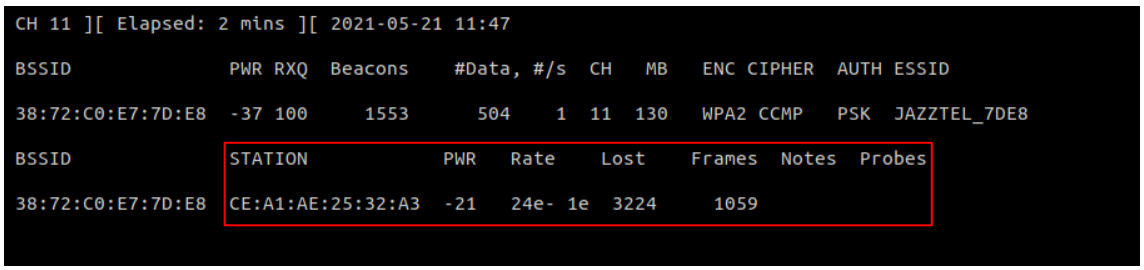


Figura 61: Llista dels clients connectats a l'AP

- Com es pot veure a la Figura 61, només hi ha una estació connectada, cosa que és normal ja que és un AP específic que només hi hem connectat el dispositiu que farà de víctima (el meu mòbil). Ara ja sabem que l'@MAC de la víctima és: CE:A1:AE:25:32:A3. Amb aquesta informació ja podem dur a terme l'atac.

- Per dur a terme l'atac cal posar la comanda de la Figura 62, on indiquem el nombre de trames a enviar (nosaltres n'enviarem 64), l'@MAC de l'AP, l'@MAC de la víctima i la interfície des de la qual realitzarem l'atac (wlo1mon).

```
mfarreras@tractor:~$ aireplay-ng --deauth 64 -a 38:72:C0:E7:7D:E8 -c CE:A1:AE:25:32:A3 wlo1mon
```

Figura 62: Comanda per a realitzar l'atac

Després d'aplicar la comanda de l'atac, es pot observar com la víctima (en aquest cas el meu mòbil) es desconnecta de la xarxa immediatament.

Durant la realització de l'atac, s'ha estat capturant el trànsit (tant el normal com el de l'atac) mitjançant l'aplicació wireshark. A la Figura 63 es pot veure una petita part de les captures realitzades, on es veu com s'envien un subconjunt dels paquets de des autenticació cap a la víctima.

```
1037 15.953482896  Apple_de:c2:14       Broadcast           802.11    312 Beacon frame, SN=3104, FN=0, Flags=........C, BI=100, SSID=d-208
1038 15.954383740  ce:a1:ae:25:32:a3    Comtrend_e7:7d:e8   802.11     38 Deauthentication, SN=23, FN=0, Flags=........
1039 15.957586152  Comtrend_e7:7d:e8    ce:a1:ae:25:32:a3   802.11     38 Deauthentication, SN=24, FN=0, Flags=........
1040 15.959789651  ce:a1:ae:25:32:a3    Comtrend_e7:7d:e8   802.11     38 Deauthentication, SN=25, FN=0, Flags=........
1041 15.963000159  Comtrend_e7:7d:e8    ce:a1:ae:25:32:a3   802.11     38 Deauthentication, SN=26, FN=0, Flags=........
1042 15.965181346  ce:a1:ae:25:32:a3    Comtrend_e7:7d:e8   802.11     38 Deauthentication, SN=27, FN=0, Flags=........
1043 15.966175761  Comtrend_e7:7d:e8    ce:a1:ae:25:32:a3   802.11     39 Deauthentication, SN=6, FN=0, Flags=........
```

Figura 63: Exemple de les captures on es veu l'atac

El problema ve quan no es poden convertir aquestes captures a un fitxer ".csv" amb tots els atributs Wi-Fi. El wireshark deixa exportar les captures a ".csv" però és un format molt simplificat que no serveix per passar-lo ni a l'script de preprocessament ni al model.

9. Conclusions

9.1. Conclusions

9.1.1. Conclusions personals

Estic molt content d'haver realitzat aquest treball, no només per tot el que après sinó per la possibilitat de poder aplicar el que ja sabia en un entorn "real". He pogut aplicar i consolidar els coneixements de xarxes que ja tenia al mateix temps que aprenia i ampliava en gran manera els meus coneixements bàsics sobre intel·ligència artificial.

També estic molt content d'haver pogut realitzar aquest projecte en un grup de recerca de la Universitat. Era el primer cop que treballava en un projecte de recerca i la veritat és que ho considero una experiència molt enriquidora tant a nivell acadèmic com a nivell personal.

Pel que fa als temes sobre els quals he aprofundit en aquest treball, m'he sorprès de la quantitat de processos que es duen a terme abans, durant i després de l'intercanvi d'informació d'una xarxa Wi-Fi. Ja coneixia per sobre els protocols i tipus de trames, però mai m'imaginava que pogués arribar a ser tan complex. Tampoc imaginava que hi hagués tantes maneres d'atacar les xarxes Wi-Fi basant-se en vulnerabilitats dels seus protocols. Respecte a la intel·ligència artificial, tot i haver tocat el tema molt per sobre i sabent que era un àmbit molt complex m'ha sorprès les possibilitats que ofereix i el seu nivell de complexitat.

9.1.2. Conclusions tècniques

En relació al grau de compliment dels objectius, en global estic content. Tot i no haver pogut completar l'objectiu final, el de provar el sistema en un escenari real, crec que s'ha realitzat una bona feina. De la resta d'objectius sí que crec que s'han complert: He après de Deep Learning i *fastai,* he aprés sobre els diferents protocols Wi-Fi i els seus atacs, he seleccionat l'atac més adequat per al sistema de detecció, he analitzat i filtrat el dataset i he implementat el sistema de detecció d'atacs de desautenticació, que ha resultat tenir una bona precisió per a detectar paquets provinents d'atacs de des autenticació.

S'ha de dir però, que tot i haver fet reajustaments al codi per tal d'obtenir unes millors mètriques, no s'ha pogut solucionar el fet que hi segueixi havent tants falsos negatius. Aquests podrien ser originats en el fet que entre els paquets del tipus "no deauthentication" hi ha altres atacs que poden tenir funcionaments semblants al d'un atac de desautenticació, provocant que el sistema es "confongui".

Per la resta, el sistema ha complert les expectatives, ja que a part d'obtenir bons resultats a la resta de mètriques, no seria gaire difícil d'extrapolar a un altre tipus d'atac (només caldria pensar quins atributs són els importants per a detectar-lo i canviar l'script de preprocessament perquè tingui en compte aquests atributs, la resta funcionaria igual). També es realitza l'entrenament en un temps molt acceptable a nivell de computació.

9.2. Problemes trobats

9.2.1. Dificultats per entendre el dataset AWID2

Tot i ser un dataset molt complet i amb una bona documentació dels diferents atacs, hi ha aspectes sobre els quals no donen informació i que podrien ser d'ajuda.

Per exemple, a la documentació del dataset mencionen que s'han de netejar els atributs per a quedar-se amb els necessaris per a detectar els atacs i evitar un mal entrenament, però no diuen quins són aquests atributs, ni com els trien. Tampoc indiquen quins són els noms dels atributs associats a cada columna, ja que els has de "deduir" tu.

Per tal d'aprendre més sobre aquesta qüestió, tant jo com en Miquel Farreras vam contactar amb els autors del dataset. En el meu cas, vaig obtenir resposta per part d'un dels coautors indicant que passaria al meu dubte a l'autor original, i també em va dir que si no rebia resposta tornés a contactar amb ell. Al cap d'unes setmanes vaig tornar a contactar amb ell (al no rebre resposta) però tot i que em va dir que tornaria a intentar contactar amb l'autor original, no vaig rebre cap més resposta.

Al final, com que es va trobar la solució a una altra banda no es va insistir més, però per part dels autors del dataset no es va rebre més ajuda.

Per a solucionar aquest problema vaig estar buscant per diversos fòrums d'internet fins que vaig trobar el github d'en brandon [14], que em va anar molt bé per definir el camí a seguir i poder veure com preprocessar les dades del dataset AWID2 correctament per al que necessitava.

9.2.2. El dataset de test és corrupte

Durant les fases inicials del desenvolupament del sistema de detecció, després d'entrenar al model i a l'intentar realitzar-hi les proves sortia un error una mica estrany. Després d'estar buscant l'error per el fòrum de *fastai* o altres fòrums similars per internet i provar diverses opcions (pe., importar alguna llibreria extra, canviar la manera de llegir les dades, canviar la manera de realitzar les proves, etc.), vaig descobrir que l'error venia del dataset que s'estava utilitzant per a realitzar les proves. El dataset en qüestió era el dataset de test que donava l'AWID2, el qual tenia una fila que donava els problemes. Un cop identificada la fila en qüestió, es va comprovar si hi havia alguna cosa anormal, però no es va trobar res, també es va provar d'eliminar la fila, però tampoc va anar bé.

Com a solució inicial, es va provar d'agafar una part del dataset de test que contingués un subconjunt de les files d'abans de la fila que provocava l'error. Això no va funcionar degut a que aquest conjunt era bastant petit i no hi havia cap fila de la classe "desauthentication", per tant no es podia testejar res amb aquest subconjunt.

La solució final vas ser (com s'ha vist) utilitzar el mateix dataset d'entrenament tant per a entrenament com per a validació, reservant un 20% del dataset a l'hora d'entrenar-lo per a validar després amb dades que no hagi vist. Posteriorment, a l'hora de realitzar les diferents proves es va utilitzar una còpia del dataset d'entrenament.

9.2.3. Problemes al convertir captures de wireshark a fitxers ".csv"

Un cop realitzat l'atac real a l'entorn del laboratori i obtingudes les captures de wireshark corresponents en format ".pcap", tocava convertir aquestes captures en un fitxer ".csv" on hi hagués tots els 154 atributs Wi-Fi per després tractar-lo amb l'script de preprocessament, passar-lo al sistema de detecció d'intrusions i comprovar el seu funcionament en un cas real.

El problema ve a l'intentar convertir les captures en format ".pcap" a ".csv" mantenint els 154 atributs Wi-Fi. Per fer això, inicialment es va intentar utilitzar l'opció de la que disposa el wireshark d'exportar en format ".csv", aquesta opció no servia perquè el fitxer resultant estava en un format molt limitat i no tenia pràcticament cap dels atributs Wi-Fi que interessen.

Per intentar solucionar-ho es va mirar al paper dels creadors de l'AWID2 [7] si explicaven com ho havien fet per passar les dades al format adequat. Hi ha una referència a la bibliografia que teòricament porta al codi que s'ha utilitzat per a obtenir els fitxers en el format desitjat, la qual està desactualitzada, ja que porta a una pàgina que no funciona. També es van intentar buscar altres opcions però com que aquesta part era a la fase final del treball i quedava poc temps, es va decidir prioritzar acabar bé la resta del treball i si després sobrava temps intentar solucionar-ho, cosa que malauradament al final no ha sigut possible.

10. Agraïments

En primer lloc m'agradaria donar les gràcies a en Lluís Fàbrega, no només per oferir-me la possibilitat de dur a terme aquest projecte sinó per tota la dedicació i l'ajuda que m'ha proporcionat, per guiar-me i orientar-me al llarg del desenvolupament del projecte.

Donar les gràcies també a en Miquel Farreras, per ajudar-me sempre que ho he necessitat i per proporcionar-me els recursos que he necessitat en tot moment.

Agrair també a en Pere Vilà que, tot i no ser el tutor del projecte, m'ha donat algun consell a l'hora de realitzar la memòria i estructurar el treball.

A més, donar les gràcies en general al grup de recerca BCDS que m'han permès desenvolupar el projecte amb molta llibertat i m'han tractat com un més del grup en tot moment.

11. Treball futur

El camp de la intel·ligència artificial és un camp molt extens amb moltes coses per aprendre i millorar on hi ha moltes opcions a realitzar com a treball futur.

Com a futur més immediat el primer que voldria fer és acabar la part de provar el sistema en un escenari real que no he pogut acabar en aquest projecte, investigant més sobre com convertir els fitxers ".pcap" al format ".csv" desitjat. Si això funcionés bé, seria interessant intentar millorar el sistema perquè actués en temps real dins una xarxa, on capturés el trànsit al moment, el transformés en el format adequat, i el passés al sistema de detecció d'intrusions perquè intentes detectar si s'està produint un atac en temps real.

També com a treball futur immediat seria interessant millorar el sistema perquè fos capaç de detectar més atacs simultàniament.

Una altra cosa a fer seria plantejar més a fons l'estructura del sistema de detecció per tal de millorar-ne la precisió i sobretot solucionar el fet que surtin tants falsos negatius.

Com a treball més llunyà seria realitzar un anàlisi a fons del dataset AWID2 per tal de triar manualment els atributs que són importants per a detectar els diferents atacs en comptes d'agafar els que fa servir en brandon [14].

Finalment, més enllà d'això es podria explorar més a fons el Deep Learning i provar si amb altres llibreries hi hauria més opcions de personalització.

12. Bibliografia

[1] Parlament europeu, ¿Qué es la inteligencia artificial y cómo se usa?. Consulta: 10 de juny de 2021. Disponible a:

https://www.europarl.europa.eu/news/es/headlines/society/20200827STO85804/que-es-la-inteligencia-artificial-y-como-se-usa

[2] Kwang Gi Kim, "Deep Learning". Consulta: 10 de juny de 2021. Disponible a:

https://synapse.koreamed.org/upload/SynapseData/PDFData/1088HIR/hir-22-351.pdf

[3] Wikipedia, "Machine Learning". Consulta: 10 de juny de 2021. Disponible a:

https://en.wikipedia.org/wiki/Machine_learning

[4] Ciro Donalek, "Supervised and Unsupervised Learning". Consulta: 10 de juny de 2021. Disponible a:

https://sites.astro.caltech.edu/~george/aybi199/Donalek_classif1.pdf

[5] Xataka, "Machine Learning i Deep Learning". Consulta: 10 de juny de 2021. Disponible a:

https://www.xataka.com/robotica-e-ia/machine-learning-y-deep-learning-como-entender-las-claves-del-presente-y-futuro-de-la-inteligencia-artificial

[6] Wikipedia, "Artificial neural network". Consulta: 10 de juny de 2021.

Disponible a:

https://en.wikipedia.org/wiki/Artificial_neural_network

[7] Constantinos Kolias, Georgios Kambourakis, Angelos Stavvrou i Stefanos Gritzalis, "Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats ana a Public Dataset". Consulta: 10 de juny de 2021. Disponible a:

https://icsdweb.aegean.gr/awid/draft-Intrusion-Detection-in-802-11-Networks-Empirical-Evaluation-of-Threats-and-a-Public-Dataset.pdf

[8] Wikipedia, Vector de inicialización (IV). Consulta: 10 de juny de 2021. Disponible a:

https://es.wikipedia.org/wiki/Vector_de_inicializaci%C3%B3n

[9] Rupinder Cheema, Divya Bansal, Dr. Sanjeev Sofat, "Implementation and security in Wireless Mesh Networks". Consulta: 10 de juny de 2021.

Disponible a:

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.259.2706&rep=rep1&type=pdf

[10] Fedora magazine, "Troubleshooting PMF for more sevure WiFi on Fedora 28. Consulta: 10 de juny de 2021. Disponible a:

https://fedoramagazine.org/troubleshoot-pmf-f28/

[11] Fastai, Making neural nets uncool again. Consulta: 10 de juny de 2021. Disponible a:

https://www.fast.ai/

[12] AWID 2, University of the AEGEAN. Consulta: 10 de juny de 2021. Disponible a:

https://icsdweb.aegean.gr/awid/awid2

[13] IEEE Xplore, Detection of De-authentication Denial of Service attack in 802.11 networks. Consulta: 10 de juny de 2021. Disponible a:

https://ieeexplore.ieee.org/document/6726015

[14] Brandon Marlowe ("Bee-Mar"-github), AWID-Intrusion-Detection. Consulta: 10 de juny de 2021. Disponible a:

https://github.com/Bee-Mar/AWID-Intrusion-Detection

[15] Ask WireShark, what is the difference between frame.time_delta and frame.time_delta_displayed? Consulta: 10 de juny de 2021. Disponible a:

https://ask.wireshark.org/question/8555/what-is-the-difference-between-frametime_delta-and-frametime_delta_displayed/

[16] WireShark, Radiotap Capture Header. Consulta: 10 de juny de 2021. Disponible a:

https://www.wireshark.org/docs/dfref/r/radiotap.html

[17] WiFi professionals, Wireshark Display Filters. Consulta: 10 de juny de 2021. Disponible a:

https://www.wifi-professionals.com/2019/03/wireshark-display-filters

[18] WireShark, Display filtre reference: wireless LAN. Consulta: 10 de juny de 2021. Disponible a:

https://www.wifi-professionals.com/2019/03/wireshark-display-filters

[19] CnofusedCoders – Nikiti Sharma, Hot to apply Deep Learning on tabular data with FastAi. Consulta: 10 de juny de 2021. Disponible a:

https://confusedcoders.com/data-science/deep-learning/how-to-apply-deep-learning-on-tabular-data-with-fastai

[20] walkwithfastai, Tabular Binary Classification (Beginner). Consulta: 10 de juny de 2021. Disponible a:

https://walkwithfastai.com/tab.clas.binary

[21] fastai, Tabular learner. Consulta: 10 de juny de 2021. Disponible a:

https://docs.fast.ai/tabular.learner.html

[22] Diyorbek Juraev-Medium, Deauthentication using the Aircrack suite in KaliLinux. Consulta: 10 de juny de 2021. Disponible a:

https://medium.datadriveninvestor.com/deauthentication-using-the-aircrack-suite-in-kalilinux-cdd13568c2a1?gi=248deba7cb95

[23] Wonder How To, Detect Script-Kiddie Wi-Fi Jamming with Wireshark. Consulta: 10 de juny de 2021. Disponible a:

https://null-byte.wonderhowto.com/how-to/detect-script-kiddie-wi-fi-jamming-with-wireshark-0186138/

[24] Wikipedia, IEEE 802.11w. Consulta: 10 de juny de 2021. Consulta: 10 de juny de 2021. Disponible a:

https://ca.wikipedia.org/wiki/IEEE_802.11w

[25] Blogspot, CCIE Wireless. Consulta: 10 de juny de 2021. Consulta: 10 de juny de 2021. Disponible a:

http://wirelessccie.blogspot.com/2016/01/80211w-aka-pmf.html

13. Annexos

13.1. 802.11w: Protected management frames

El protocol 802.11w és una modificació de l'estàndard 802.11 que incrementa la seguretat de les trames de gestió [24].

De les trames de gestió que protegeix destaquen les trames de dissociació i les trames de desautenticació. Per altra banda, de les trames que no protegeix destaquen les d'autenticació, les beacon, les de probe request i les de probe response.

L'objectiu d'aquesta millora és el de protegir trames que puguin resultar en problemes en cas de ser suplantades, com és el cas de les trames de desautenticació (com s'ha vist en aquest treball) [25].

Aquesta modificació només s'aplica a xarxes sense fils que disposin de xarxes de seguretat robustes (Robust Security Networks o RSN), com és el cas de xarxes amb WPA2. Al activar les PMF en una xarxa d'aquestes condicions s'afegeix un hash (o firma) anomenat MIC (Message integrity Code) a diverses trames de gestió mentre l'estació es mantingui connectada a la xarxa. Això provocarà que si un atacant intenta suplantar alguna d'aquestes trames no disposarà del hash per tal de "firmar" les trames, provocant que tant l'AP com les altres estacions ignorin la trama suplantada ja que no estarà ben "firmada" (Figura 64).

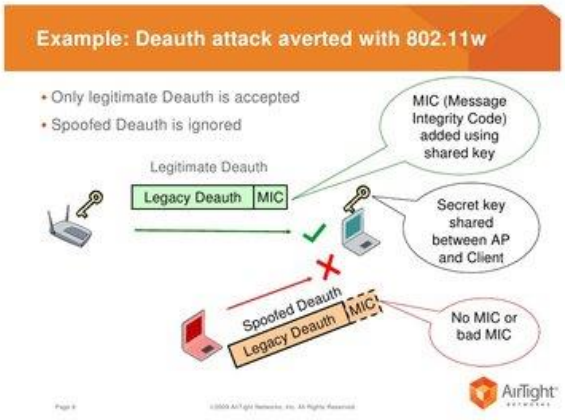


Figura 64: Exemple d'atac en 802.11w

13.2. Codi font

13.2.1. Script de preprocessament

```
from pathlib import Path

import numpy as np
import pandas as pd

# Create a path to the 'resources' directory in our repository
resource_dir = Path('./', 'resources')

# Create a list of columns that want to keep in our analysis (from Dr.
Chung's webpage)
desired_cols = [2, 5, 45, 62, 64, 65, 68, 71, 74,
                75, 88, 91, 92, 105, 106, 110, 116, 120, 154]

# Specifically read in our desired columns from the reduced AWID
dataset
data = pd.read_csv(
    Path(resource_dir, 'AWID-ATK-R-Trn.zip'),
    sep=',',
    compression='zip',
    usecols=desired_cols)

col_names = []

# Iterate over the text file containing the column names,
# if the line number matches one of our desired columns, add that name
to a list
with open(Path(resource_dir, 'col_names.txt')) as cols:
    for line_num, col_name in enumerate(cols):
        if line_num in desired_cols:
            col_names.append(col_name.rstrip())

# Set the column headers to the names from the Wireshark frame
data.columns = col_names

# Replace the '?' string in the Pandas DataFrame with a NumPy NaN
value
data.replace('?', np.nan, inplace=True)

# If over 60% of the values in a column is NaN, remove that column
# because it is not useful for our analysis
prev_num_cols = len(data.columns)
data.dropna(axis='columns', thresh=len(data.index) * 0.40,
inplace=True)
print("Removed " + str(prev_num_cols - len(data.columns)) +
      " columns with all NaN values.")

# If there are no values or only one unique value in a column,
# remove that column because it is not useful for
# distinguishing between normal and attack type
cols_to_drop = []

for col in data:
    if not data[col].nunique() > 1:
        cols_to_drop.append(col)

data.drop(columns=cols_to_drop, inplace=True)
```

```
print("Removed " + str(len(cols_to_drop)) +
      " columns with no variation in its values.")

# Drop the rows that have at least one NaN value in it
old_num_rows = data.shape[0]
data.dropna(inplace=True)
print("Removed " + str(old_num_rows -
                       data.shape[0]) + " rows with at least one NaN
value in it.")

# Change the name of normal frames and all the frames that aren't
"deauthentication" to "no deauthentication"
#data["class"] = data["class"].replace(['amok', 'arp',
'authentication_request', 'beacon', 'cafe_latte', 'evil_twin',
'fragmentation', 'normal', 'probe_response'], 'no deauthentication')
data["class"] = data["class"].replace(['amok', 'arp', 'beacon',
'cafe_latte', 'evil_twin', 'fragmentation', 'normal',
'probe_response'], 'no deauthentication')

# Export the pandas DataFrame as a CSV file
data.to_csv(
    Path(resource_dir, 'preproc_dataset-un-sol-atac.csv'),
    index=False,
    sep=',')
```

13.2.2. Sistema de detecció d'intrusions

Detect_Wi-FI_Attacks

June 10, 2021

1 Detecció d'atacs Wi-Fi mitjançant Deep Learning

[1]:
```
from fastai.tabular.all import *
from fastai import *
```

1.1 Lectura i tractament de les dades

Agafem el dataset d'entrenament:

[2]:
```
train = pd.read_csv('preproc_dataset-un-sol-atac.csv')
train.head()
```

[2]:

	frame.time_delta_displayed	radiotap.flags.shortgi	radiotap.rxflags.badplcp	wlan.fc.type	wlan.fc.frag	wlan.fc.moredata	wlan.duration	wlan.ra	wlan.fcs_good	class
0	0.024271	0	0	0	0	0	0	ff:ff:ff:ff:ff:ff	1	no deauthentication
1	0.001631	0	0	0	0	0	0	ff:ff:ff:ff:ff:ff	1	no deauthentication
2	0.055325	0	0	0	0	0	0	ff:ff:ff:ff:ff:ff	1	no deauthentication
3	0.000415	0	0	2	0	0	44	28:c6:8e:86:d3:d6	1	no deauthentication
4	0.000005	0	0	1	0	0	0	00:25:bc:ed:07:cf	1	no deauthentication

Agafem el dataset de test

[3]:
```
test = pd.read_csv("preproc_dataset-un-sol-atac-test.csv")
```

Definim la variable dependent (sobre la que volem fer la predicció)

1

[4]:
```
var_dependent = 'class'
```

Definim els noms de les variables categòriques:

[5]:
```
var_cat = ['wlan.ra','radiotap.flags.shortgi', 'radiotap.rxflags.badplcp'␣
↪,'wlan.fc.type','wlan.fc.frag' ,'wlan.fc.moredata',  'wlan.fcs_good']
```

Definim els noms de les variables contínues

[6]:
```
var_cont = ['frame.time_delta_displayed', 'wlan.duration']
```

Definim els processos que volem aplicar a les dades

[7]:
```
procs = [Categorify, FillMissing, Normalize]
```

Definim les particions

[8]:
```
splits = RandomSplitter(valid_pct=0.2)(range_of(train))
```

Construïm un objecte mitjançant el TabularPandas

[9]:
```
to = TabularPandas(train, procs = procs, cat_names = var_cat, cont_names =␣
↪var_cont, y_names = var_dependent, splits = splits)
```

Passem l'objecte anterior a un dataloader

[10]:
```
dls = to.dataloaders(bs=4096)
```

[11]:
```
dls.show_batch()
```

```
<IPython.core.display.HTML object>
```

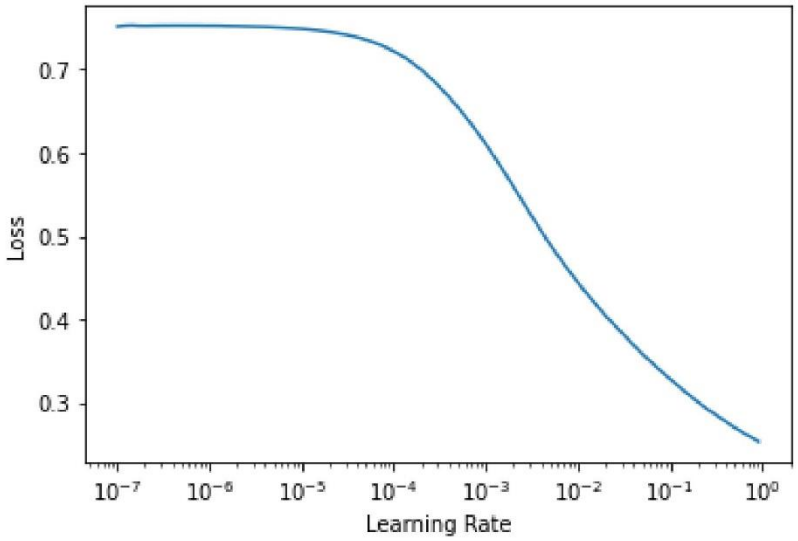
1.2 Definim el nostre model creant un learner

[12]:
```
learn = tabular_learner(dls, [200, 100], metrics=[accuracy])
```

[13]:
```
learn.lr_find()
```

```
<IPython.core.display.HTML object>
```

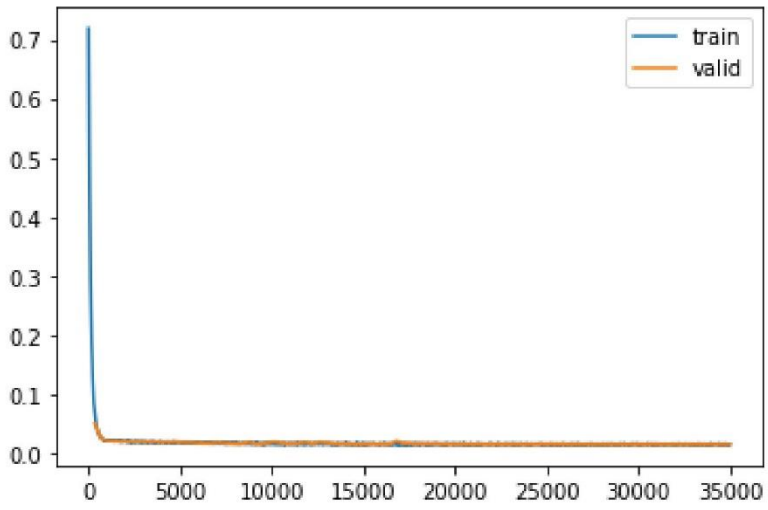
[13]:
```
SuggestedLRs(lr_min=0.09120108485221863, lr_steep=0.0020892962347716093)
```



[14]:
```
learn.fit_one_cycle(100)
```

```
<IPython.core.display.HTML object>
```

[15]:
```
learn.recorder.plot_loss()
```



3

[16]:
```
learn.show_results()
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

1.3 Provem les prediccions amb el dataset de test

Fem les prediccions

[17]:
```
test = train.copy()
```

[18]:
```
dl = learn.dls.test_dl(test, bs=4096)
```

[19]:
```
dl.show_batch()
```

```
<IPython.core.display.HTML object>
```

[20]:
```
preds, _ = learn.get_preds(dl=dl)
```

```
<IPython.core.display.HTML object>
```

[21]:
```
final_preds = preds.numpy()
```

[22]:
```
final_preds = np.argmax(final_preds, axis=1)
```

[23]:
```
final_preds
```

[23]:
```
array([1, 1, 1, …, 1, 1, 1])
```

[24]:
```
comptador_deauth = 0
comptador_correctes_deauth = 0
comptador_incorrectes_deauth = 0
fila = 0
for x in final_preds:
    if x == 0:
        real = train.iloc[fila][9]
        if real == "deauthentication":
            comptador_correctes_deauth += 1
        else:
            comptador_incorrectes_deauth += 1
        comptador_deauth += 1
    fila += 1
```

```
print("Total: ", comptador_deauth, "Precisió: ", (comptador_correctes_deauth/
↪comptador_deauth)*100)
```

```
Total:  3056 Precisió:  70.6151832460733
```

[25]:
```
comptador_no_deauth = 0
comptador_correctes_no_deauth = 0
comptador_incorrectes_no_deauth = 0
fila = 0
for x in final_preds:
    if x == 1:
        real = train.iloc[fila][9]
        if real == "no deauthentication":
            comptador_correctes_no_deauth += 1
        else:
            comptador_incorrectes_no_deauth += 1
        comptador_no_deauth += 1
    fila += 1
print("Total: ", comptador_no_deauth, "Precisió: ",␣
↪(comptador_correctes_no_deauth/comptador_no_deauth)*100)
```

```
Total:  1790546 Precisió:  99.53706858131542
```

[26]:
```
real_deauthentication = train.isin(['deauthentication']).sum(axis=0)
```

[27]:
```
real_no_deauthentication = train.isin(['no deauthentication']).sum(axis=0)
```

[28]:
```
real_deauthentication[9]
```

[28]:
```
10447
```

[29]:
```
real_no_deauthentication[9]
```

[29]:
```
1783155
```

[30]:
```
print("Positius: ", comptador_correctes_deauth, "Realment positius: ",␣
↪real_deauthentication[9])
print("Negatius: ", comptador_correctes_no_deauth, "Realment negatius: ",␣
↪real_no_deauthentication[9])
print("Falsos positius: ", comptador_incorrectes_deauth)
print("Falsos negatius: ", comptador_incorrectes_no_deauth)
```

```
Positius:  2158 Realment positius:  10447
Negatius:  1782257 Realment negatius:  1783155
Falsos positius:  898
Falsos negatius:  8289
```