Interception des ondes radios sur les capteurs d'énergies

05.04.2021

—

Deep Enstamina

Maxence Brugères (Telecom Paris & Sciences Po)

Rôle : Classification random forest, outil de séparation de trames en touche, outil de production de mots

Clément Bernard (Telecom Paris & Polytechnique Montréal)

Rôle : Classification GRU, MLP, outil de séparation des touches du login et du mot de passe, mco du répo Github

Inès Benito (ENSTA Bretagne)

Rôle : Classification via d'autre modèle (Logistic regression, KNN), recherches théoriques et appliquées pour entrainer des modèles avec des loss custom, synthèse des travaux pour le rapport, data visualisation

Corentin Lestrat (ENSTA Bretagne)

Rôle : État de l'art des exploitations de SPC avec du machine learning, recherche biblio sur des sujets divers en support sachant qu'il n'avait aucune formation technique en machine learning, social engineering

Introduction

Un ingénieur de la DGA-MI vient de se loger sur son poste de travail. Tant que faire se peut, nous avons essayé de nous placer dans la peau d'une personne malveillante ayant accès aux signaux parasites émis par son clavier.

L'objectif est donc de trouver ce login et ce mot de passe. Notre travail est de développer différents outils pour permettre et automatiser la détection des touches pressées sur ce poste de travail de la DGA-MI. Il s'agit :

- d'un outil isolant les trames qui correspondent à des pressions de touches différentes
- d'un outil séparant les trames du login et celles du mot de passe
- de modèles de machine learning qui estiment les touches pressées ou leur probabilité
- d'un outil permettant de générer les logins et mots de passe les plus probables afin de réduire au maximum le temps nécessaire au bruteforce du login/mot de passe.

Analyse des données et statistiques

Pour commencer, nous avons tout d'abord étudié les signaux parasites générés lors de la pression d'une touche du clavier. Ces données correspondent à des ondes radios sous forme de trames de 17 pics variant en fonction de la touche de clavier pressée. D'abord, un ensemble de données connues : 41 touches ont été pressées, nous donnant une idée de l'allure des pics pour chacune de ces touches. Il est également fourni un ensemble de trames inconnues, dont le matching avec les touches pressées n'est pas fait. Ces trames correspondent au déverrouillage d'un ordinateur, par l'entrée de touches CTRL + ALT + SUPPR, puis d'un login et enfin d'un mot de passe.

I. Etude des pics

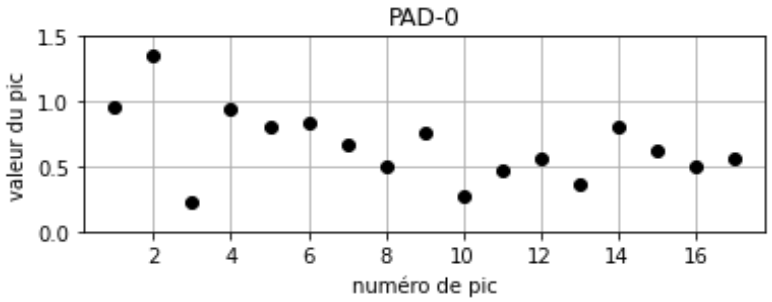


Figure 1 - Exemple d'une trame de 17 pics lorsque la touche 0 est pressée

L'observation des trames de pics permet d'intuiter que la distribution des pics diffère suivant la touche pressée. Pour mettre cela en avant, nous avons étudié la distribution statistique de chacun des 17 pics pour chacune des 42 touches enregistrées. Cette étude est disponible sur le répertoire github. Un exemple de cette étude statistique pour le pic numéro 6 est disponible ci-dessous. Ces représentations sont des boîtes à moustache qui donnent les premier et troisième quartiles, les valeurs extrêmes ainsi que la médiane. On remarque que les distributions statistiques de certains pics sont caractéristiques de la touche pressée. Par exemple, le troisième pic d'une trame de 17 peut permettre d'identifier si la touche pressée est un nombre entre 1 et 4. De même, la distribution statistique du 6ème pic d'une trame semble être différenciante pour la touche SHIFT. On remarque

que pour le 6ème pic, la touche SHIFT a une boîte à moustache qui se démarque des autres avec une moyenne plus élevée.

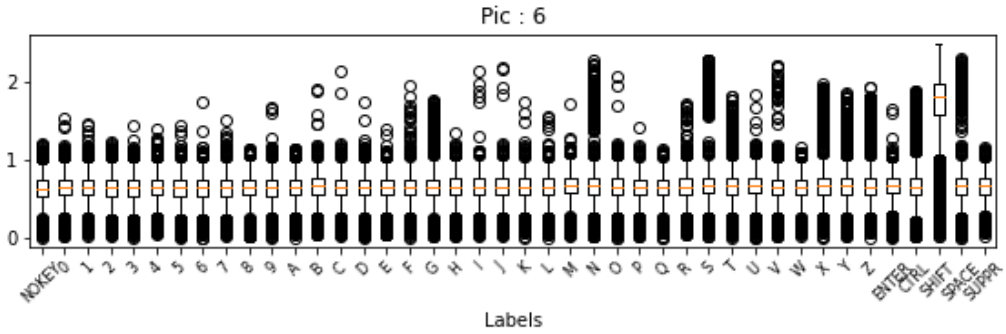
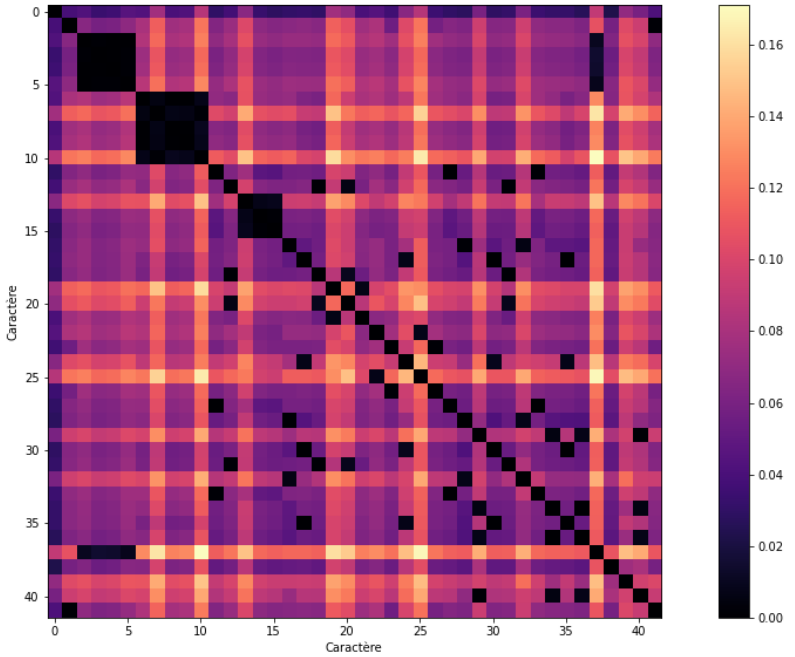


Figure 2 - Répartition statistique du 6ème pic pour chacune des touches

II. Distance entre les différents caractères

Pour évaluer la similarité des distributions statistiques pour certains pics, nous avons étudié la distance euclidienne entre les trames des différentes touches. Pour ce faire, nous calculons la trame moyenne pour chacun des 42 caractères. La figure 3 représente les distances euclidiennes entre les trames moyennes des 42 caractères, numérotées dans le même ordre que sur l'axe des abscisses de la figure 2. On observe par exemple que les caractères numéro 3 à 5, qui correspondent aux touches 1 à 4, ont une distance très faible entre eux. C'est bien ce qui était pressenti dans la partie précédente. Il est alors raisonnable de penser que ces touches seront difficiles à différencier pour les modèles de machine learning.



Labels	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
Touche pressée	NOKEY	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	ENTER	CTRL	SHIFT	SPACE	SUPPR

Figure 3 -Distance entre les trames moyennes de chacun des 42 caractères

Classification des caractères

Nous avons considéré chacun des 42 caractères comme étant une classe. De ce fait, notre problème de détection de login et mot de passe s'est transformé en classification multi-classe à 42 labels. Afin d'entraîner un classifieur sur les ondes enregistrées lors de l'authentification de l'utilisateur, nous avons utilisé chacun des enregistrements des 42 caractères comme données d'entraînement. Deux approches ont été étudiées :

- Chaque trame est classifiée dans une des 42 classes et donnée telle quelle pour l'entraînement.
- N trames sont regroupées, N étant un hyperparamètre, de telle sorte que le classifieur reçoit en entrée (N,17) données et renvoie un label.

I. Trame unique

Tout d'abord, nous avons entraîné nos réseaux avec une trame unique de 17 pics. Plusieurs pistes ont été explorées quant au choix du réseau.

A. Réseaux de neurones

Une première approche a été de considérer un réseau de neurones très simple : une seule couche avec 2048 neurones et ReLU comme fonction d'activation. L'idée étant qu'avec suffisamment de neurones dans une seule couche, le réseau va discrétiser à la main les différences entre les pics des entrées, sans traitement préalable. Cependant, cette méthode ne nous a pas permis d'avoir une précision supérieure à 56% sur l'ensemble de validation. De ce fait, nous avons exploré une méthode plus complexe pour augmenter la variance, au risque d'avoir du surapprentissage.

B. Gated Recurrent Unit

Une seconde approche a été de voir chacun des pics comme étant récurrent. De ce fait, nous avons essayé de donner un sens temporel à chacun des pics, et avons utilisé un GRU pour cela. Cette modélisation est discutable car elle introduit une nouvelle dimension temporelle, cependant elle nous a permis d'obtenir de meilleurs résultats qu'avec les réseaux de neurones précédents. Pour augmenter le nombre d'informations disponibles, nous avons ajouté une couche linéaire de taille (17,128) permettant ainsi de transférer les informations des 17 pics dans une dimension plus grande, puis avons appliqué un GRU avec 512 "hidden units".

C. Random Forest : classification avec la matrice de confusion

Lorsque l'on tente d'attribuer une touche à une trame, les caractéristiques statistiques des trames vues dans la première partie sont déterminantes et permettent d'éliminer des possibilités de touches, on peut ainsi construire un arbre de décision. C'est pour cela que nous avons opté pour un classifieur Random Forest. Les hyper-paramètres du modèle tels que le nombre d'arbres ou leur profondeur maximale sont choisis en maximisant les scores obtenus pour chaque hyper-paramètre avec de la cross-validation. Une fois les paramètres optimaux trouvés, le modèle entraîné donne des scores de précision et f1 similaires, voire un peu moins bons, aux modèles précédents.

Une fois le modèle ajusté, on classifie les trames de la séquence d'enregistrement du login et du mot de passe. Les probabilités d'appartenance de chaque trame à l'une des 42 touches connues sont présentées ci-dessous.

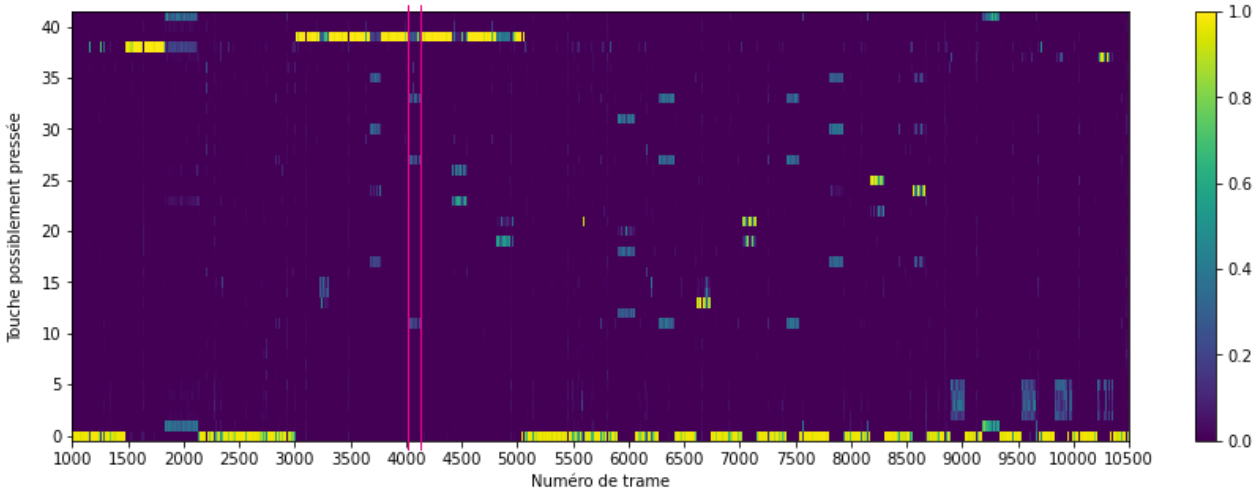


Figure 4 - Probabilité de catégorisation de chacune des trames de la séquence de login. (“Vecteurs de probabilité” de chaque trame).
Entre les deux règles roses, un exemple de vecteur de probabilité (vecteur de taille 42 représentant la probabilité d’appartenance d’une trame à chacune des 42 catégories)

On distingue très nettement que différentes touches sont pressées au fil des trames qui défilent. Cependant leur classification n’est pas franche, sinon pour chaque trame il n’y aurait qu’une touche avec une probabilité non-nulle. En effet, comme on peut le voir sur la matrice de confusion, beaucoup de classes sont mal classifiées, ceci s’explique par la faible distance euclidienne qui sépare les trames moyennes de ces classes, voir Figure 5.

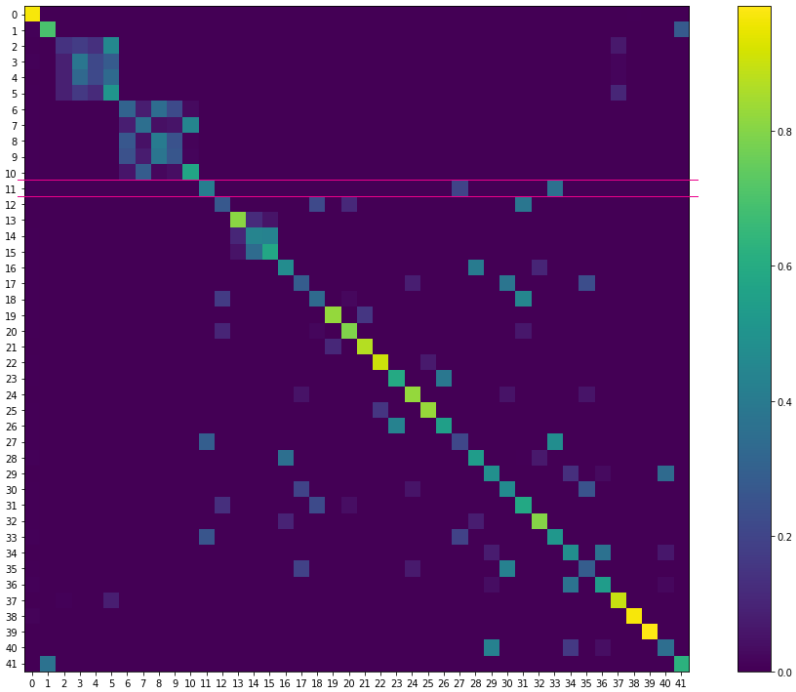


Figure 5 - Matrice de confusion du random forest **(produite via un set de données différent de celui de l'entraînement).**

Entre les deux règles roses, la ligne de la matrice de confusion qui minimise la distance avec le vecteur identifié dans la figure précédente. Il s’agissait donc sûrement de la même touche pressée, à savoir la touche n°11 qui correspond à la lettre A.

Cependant, lorsqu'on se place du point de vue de l'attaquant, chaque touche pressée générera plusieurs trames comme montré ci-dessus et on compte au moins une centaine de trames pour chaque touche pressée. On obtient un "vecteur de probabilité" de taille 42 pour chaque trame. On peut alors approximer la probabilité moyenne de classification d'une touche pressée sur une centaine de trames (Fig 4) et ainsi comparer cette probabilité aux probabilités de classification des touches connues (décrites par la matrice de confusion) (Fig 5).

En d'autre termes, on peut calculer la distance entre la moyenne des vecteurs de probabilités de chaque trame pour une même touche pressée (sur une centaine de trames) et chacune des lignes de la matrice de confusion. La distance la plus faible indique ainsi la prédiction la plus probable. On notera que la distance minimale entre deux lignes de la matrice de confusion tend naturellement à augmenter lorsqu'on entraîne le modèle (le modèle parfait produirait une matrice identité).

D. Comparaison des modèles

Afin de comparer les modèles, nous avons considéré la précision et le score f1 pour mieux prendre en compte les mauvaises classifications. La Figure 6 nous montre les scores f1 pour les touches du pavé numérique, les touches autres que les chiffres et pour toutes les données :

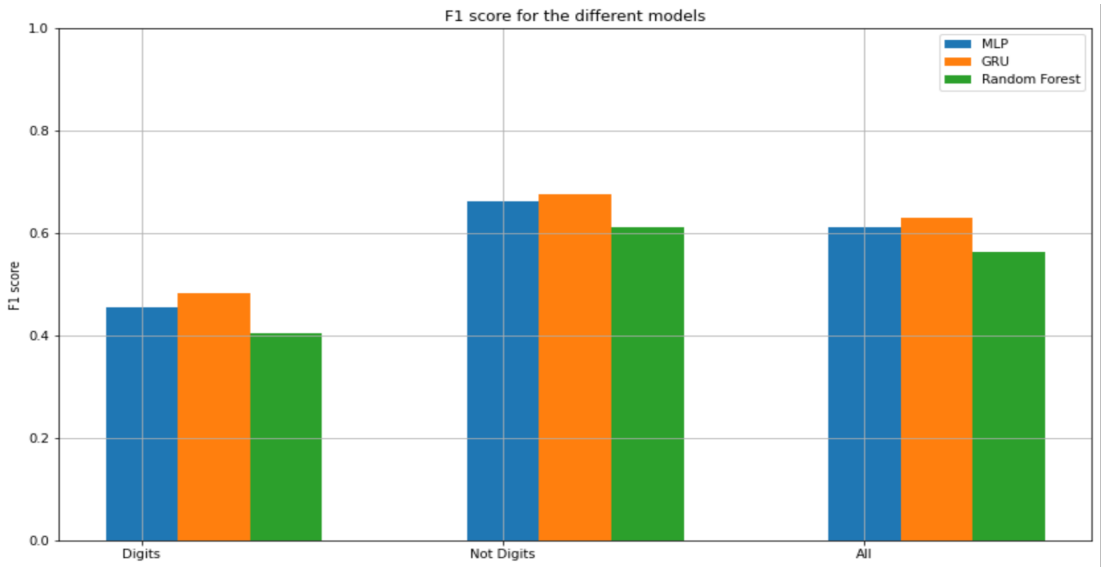


Figure 6 - Score f1 pour MLP, GRU et Random Forest sur l'ensemble de validation (pour les chiffres, autres que les chiffres et toutes les données)

On remarque que les scores obtenus avec les MLP et le GRU sont meilleurs que ceux obtenus avec random forest sur les données de validation. Cependant c'est le résultat inverse que nous obtenons sur les données de test. En effet, le MLP et le GRU font du sur-apprentissage sur les données d'entraînement qui représentent des pressions d'une unique touche. Or les données de tests peuvent présenter des pressions de plusieurs touches en même temps.

Par ailleurs, la méthode du random forest combinée aux calculs de distances à la matrice de confusion délivrent une précision de 68% lorsqu'on accepte uniquement la classe la plus probable en prédiction. Ce score augmente à 83% lorsqu'on accepte les deux classes les plus probables en réponse. C'est donc ce modèle qui est retenu puisqu'il distingue mieux les pressions de plusieurs touches à la fois.

II. Trames groupée

Comme mentionné dans la partie précédente, le scénario proposé implique que l'attaquant aura à sa disposition plus d'une centaine de trames pour chaque pression de touche lors de la saisie du login et du mot de passe. Une autre manière d'entraîner un modèle pour qu'il corresponde mieux au modèle d'attaque serait donc de l'entraîner avec des groupements de trames en entrée, et non pas avec des trames seules.

A. Sans bruit

L'idée est de grouper les trames par groupe de N, et d'entraîner un classifieur dessus. Nous avons utilisé pour cela les trois mêmes classifieurs que précédemment. La seule différence est que le GRU prend en entrée (N,17) trames, et envoie donc N fois une trame de 17 dans son encodeur avant de donner une prédiction. Pour Random Forest et le MLP, nous avons remplacé les données d'entraînement de taille 17 par des données de taille 17*N. Les résultats avec Random Forest sont assez mauvais et n'ont pas été retenus.

On obtient finalement une meilleure précision sur les phases d'entraînement et de validation en groupant les trames en entrée. Cependant le problème de sur-apprentissage persiste et ces classifieurs généralisent mal sur les données de test.

B. Ajout du bruit

Comme les modèles précédents overfit sur les données d'entraînement non-bruitées, nous avons essayé de produire des données bruitées simulant une pression de plusieurs touches. Pour ce faire, nous avons sélectionné, pour chaque label, une trame aléatoire et une trame venant d'un autre caractère (C fois, avec C un hyperparamètre égal à 100 pour l'expérience). Ensuite, nous avons, avec probabilité 0,25 chacune, ajouter les trames entre elles, fait la différence des trames, multiplié les trames termes à termes, et fait la moyenne. Comme nous ne savons pas comment interférer les signaux, nous avons essayé ce bruitage. Nous avons essayé de faire de même avec GRU et MLP avec et sans groupement de trames. Nous obtenons les résultats de la figure 7 sur l'ensemble de validation **sans bruit**

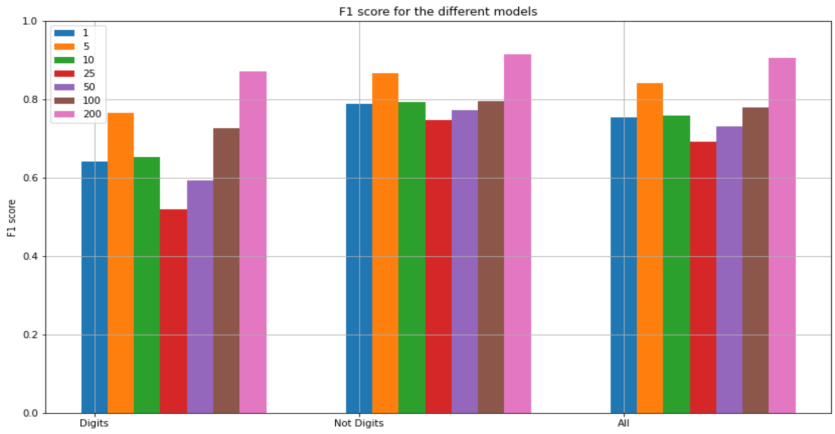


Figure 7 - F1 score pour différent groupement de trames avec GRU entrainé sur des données bruitées et testés sur des données non bruitées

On remarque que les performances sur les données bruitées sont meilleures que celles des modèles entraînés sur des données non bruitées. Par exemple, pour un groupement de 5 trames,

on obtient un score f1 proche de 0.85. Cependant, cette méthode n'est pas aussi performante sur les vraies données d'un utilisateur : le modèle fait du sur-apprentissage sur le bruit artificiellement créé, et ne généralise pas assez. Nous avons aussi essayé de rajouter du bruit uniquement sur les trames de chiffres ou les trames de SHIFT,car l'appui simultané des touches SHIFT et d'une lettre pour écrire une majuscule sont difficilement identifiables, mais les résultats n'ont pas été concluants. Nous n'avons pas réussi, dans le temps imparti, à creuser l'idée d'ajout artificiel de bruit, malgré nos innombrables idées.

Identifier le login et le mot de passe

I. Identification des trames des touches pressées

A. Identification des trames

L'attaquant veut pouvoir délimiter les trames qui appartiennent à une même touche pressée. Pour se faire, on reprend les probabilités de classification du random forest pour chaque trame. On applique ensuite une moyenne glissante sur une fenêtre de 30 trames pour lisser les probabilités. Enfin on moyenne la distance euclidienne entre chacun des vecteurs de probabilités des trames dans une fenêtre de 30 trames. Cette distance est maximale quand la fenêtre se trouve exactement sur la frontière entre deux touches différentes.

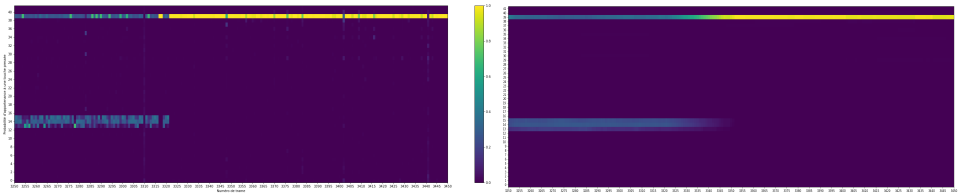


Figure 8 - Exemple de vecteurs de probabilités de classification des trames avant et après moyennage

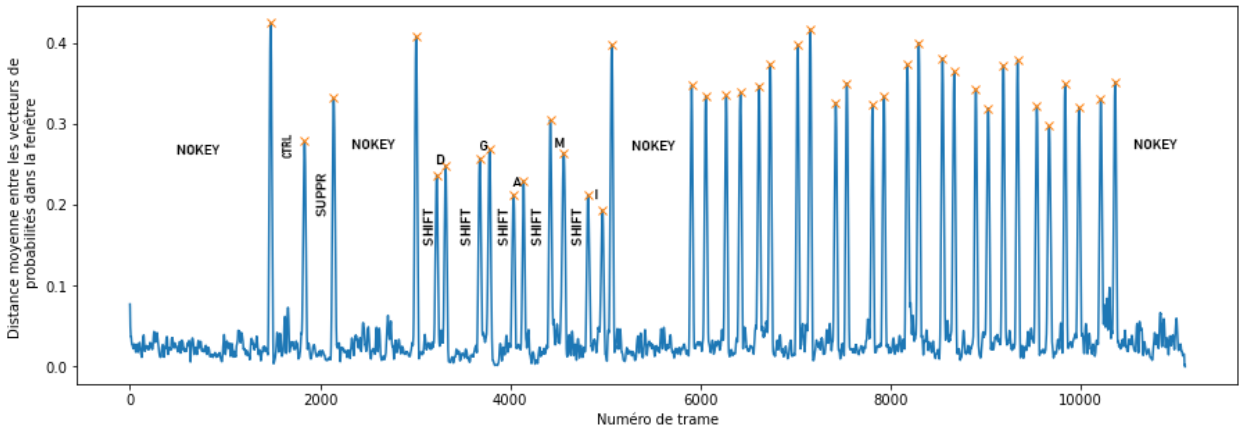


Figure 9 - Pics indiquants les changements de touches pressées

Les pics oranges sont ainsi situés sur le numéro de la trame où la touche pressée change. A noter que la détection de pics se fait via l'outil scipy et un algorithme de clustering permettant d'éliminer les "petits pics" ayant une distance moyenne inférieure à 0.1 qui sont assimilés à du bruit.

B. Identification du login et mot de passe

L'attaquant, à ce stade, a un outil pour détecter les trames correspondant à une même touche pressée. Il veut également dans une certaine mesure pouvoir distinguer les touches du login et du mot de passe, pour se faire nous avons en plus développé un **outil de clustering séparant automatiquement le login et le mot de passe** en fonction du temps séparant les frappes. Il peut être amélioré via l'ajout de la détection de la touche tabulation. On utilise ici l'algorithme "Agglomerative Clustering" qui fusionne deux éléments en minimisant la distance euclidienne entre eux. Le nombre de clusters est mis à 3 car nous considérons qu'il y a 3 groupes; un pour déverrouiller l'ordinateur, un pour le login et un pour le mot de passe. Cela se généralise uniquement si l'utilisateur rentre son login et son mot de passe dans un temps assez court.

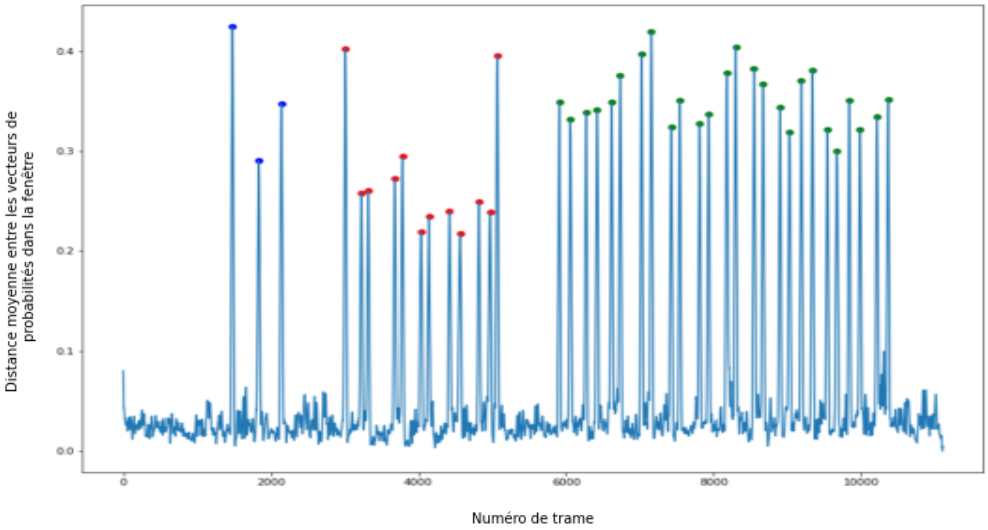


Figure 10 - Séparation des touches du CTRL+ALT+SUPPR, login et mot de passe

II. Identification des caractères

Grâce au modèle random forest entraîné et tous les outils développés précédemment, nous pouvons désormais identifier les touches pressées dans la séquence de login. Le graphique ci-dessous représente la distance entre le vecteur de probabilité moyen de la touche réellement pressée, obtenu avec random forest, et les vecteurs de probabilité moyens des touches connues. Pour rappel, les vecteurs de probabilité moyens des touches connues sont les lignes de la matrice de confusion. On peut ainsi identifier pour chaque touche pressée les 5 touches les plus probables.

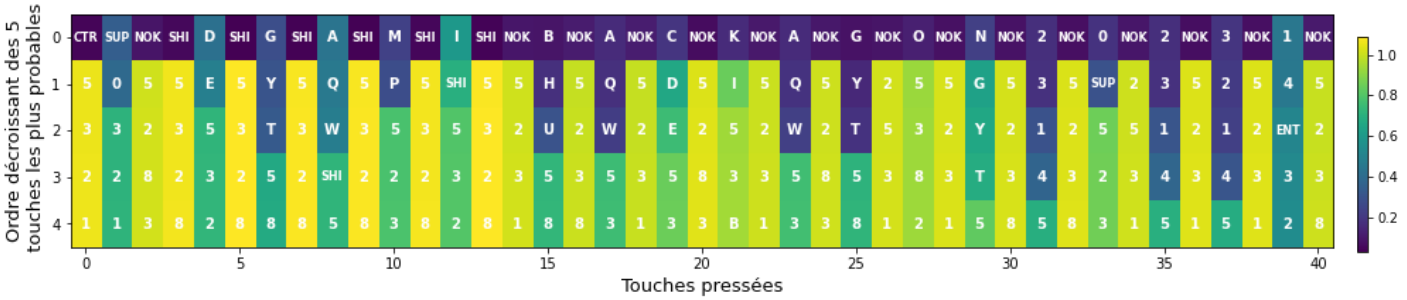


Figure 11 - Prédiction finale du login et du mot de passe

On pourra noter que les touches qui se confondent semblent la plupart du temps être physiquement proches sur le clavier.

III. Production des login et mots de passe

Enfin, dans le cadre d'une attaque par bruteforce, l'attaquant veut avoir accès à une liste des logins et des mots de passe les plus probables pour réduire au maximum le temps et le ressources nécessaires. Nous avons donc, à partir d'une matrice de distance comme montré ci-dessus, produit un outil qui génère les logins et mots de passe à la plus forte probabilité, donc qui minimise la distance totale de la séquence de touches en sortie. Cette outil fournit les cinq suites de caractères suivant comme étant les plus probables:

['CTRL+SUPPR+DGAMIbackagon20231',
'CTRL+SUPPR+DGAMIbackayon20231',
'CTRL+SUPPR+DGAMIhackagon20231',
'CTRL+SUPPR+DGAMIhackayon20231',
'CTRL+SUPPR+DGAMIbqckagon20231']

Combiné avec la fonction de clustering du login et du mot de passe, la sortie sépare les touches appartenant au CTRL + SUPPR en bleu, au login en rouge et au mot de passe en vert. Par ailleurs on notera que la dernière touche pressée semble être un chiffre mais présente une distance plus importante que les autres chiffres dans la séquence. En observant les classes probables pour cette dernière touche de la séquence, on remarque qu'il n'y a pas uniquement des chiffres mais également la touche ENTER. En sachant que les GRU et MLP tendent à classifier cette dernière en touche ENTER, nous pouvons raisonnablement penser qu'il s'agit bien d'un ENTER.

De plus, en bon hackers, nous avons fait des recherches sur la structure à hacker pour ne pas avoir à tester tous les mots de passe trouvés et grâce à du "social engineering" nous trouvons:

- Login: **DGAMI**
- Mot de passe: **hackaton2021**

Conclusion

En conclusion nous avons produit un ensemble complet d'outils permettant à un attaquant de faire une analyse en temps réel des frappes d'un clavier:

- Détection du nombre de touches pressées
- Association des trames correspondantes à ces touches
- Séparation du login et du mot de passe
- Classification des touches
- Production des logins et mots de passe les plus probables

Ces outils pourraient être améliorés en utilisant des trames correspondant à l'appui de touches en simultanées pour l'entraînement des modèles. Par exemple CTRL+ALT, CTRL+ALT+SUPPR ou encore SHIFT + une lettre. Il serait intéressant de chercher la forme du signal physique résultant de la superposition des parasites liés à ces touches s'il s'avérait impossible d'enregistrer proprement ces superpositions.