Simon Alix
Marc Crampe
Lisa Mignerey
Hugo Quin

10/01/2024
CDI

HANDS ON MACHINE LERANING FOR CYBERSECURITY

From image to malware

Proposition 3

Summary

1- How malware detection could take advantage of the typical skills of CNNs in image recognition

A- Introduction

In cybersecurity, defenders and company must be prepared to all kind of cyber threat, this includes malware threats. The best tool to respond is an update antivirus, usually it compares signature of a local application with a database of malware signature.

These technics work as long as the database is comprehensive, but more and more cyber criminal developpe new malware. The major issue is that all the classic antiviruses can't respond to it because it isn't in the database.

The goal of our study is to explore how new technologies and technics of neural networks can detect zero-day malware. For that we use the article *"Towards Building an Intelligent Anti-Malware System: A Deep Learning Approach using Support Vector Machine (SVM) for Malware Classification"*.

B- Signature

In the field of malware detection, we focus on the signature of the malware to try to detect them. We can't easily access to the source code of the virus, we only have in hand the compiled file (.exe). This file can be read as a sequence of bits, the most basics instruction for a computer. Our goal is to categorize based on the bit code if it's a malware or not and maybe its family.

C- Study paper

In the studied paper, the strategy put in place is to visualize malwares as images in order to use the advantage of Convolutional networks. To do so, the malware binary code is converted to a grayscale image:

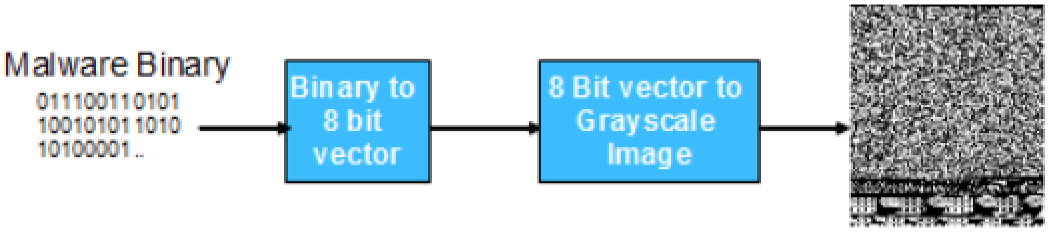


Figure 1: process of transforming binary code into image.

We recreate the code the extract image from the dataset. The images are merely differentiable to the human eye, and it is impossible to recognize patterns between images of the same family.

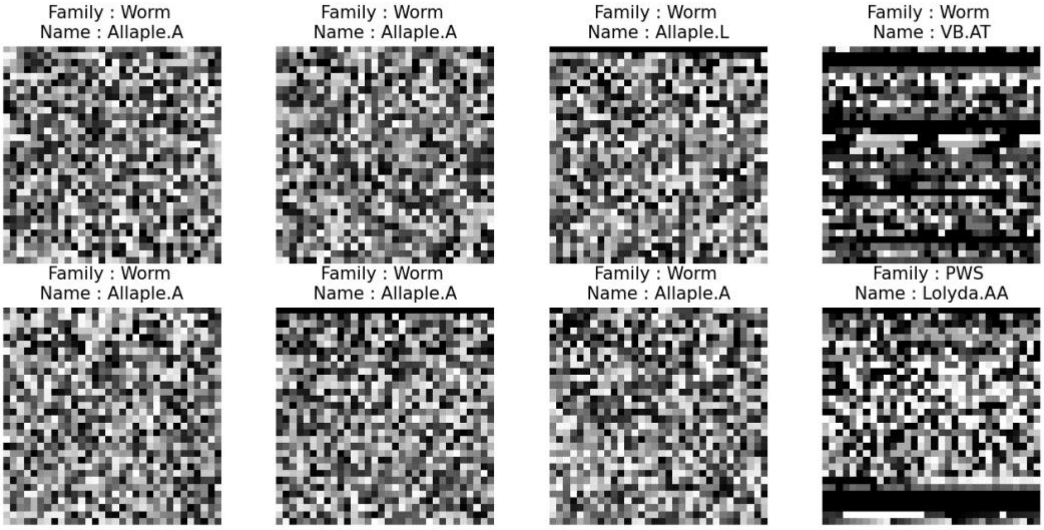


Figure 2: Malware Converted to Gray Scale Images

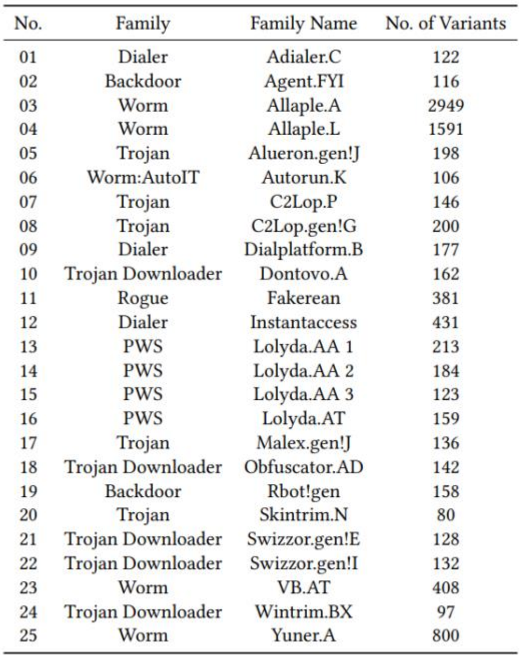
2- Usage of images for malware detection and how this could be done from images with CNNs?

A- Malware behavior

As we already said, it's almost impossible to design a strategy for preventing virus that are just released, because we only have access to the binary code. Usually, defender have to run new virus on virtual machines and trace back all the action of the one, it's called retro engineering. A long and complex task for specialist.

B- Ideal case

The intention of the paper is to present a protocol that can discrimen malware based on their associated image, this could be used to accelerate the work of the specialist by giving them the right direction to looking for. Paper's algorism is trained on a dataset *Malimg,* which study is composed of 9932 images, 2D matrixes of size 32x32, and is classified in 24 different families of malwares as we can see on this table (*figure 2*). We want to classify each malware image into one of these family of malware.



No.	Family	Family Name	No. of Variants
01	Dialer	Adialer.C	122
02	Backdoor	Agent.FYI	116
03	Worm	Allaple.A	2949
04	Worm	Allaple.L	1591
05	Trojan	Alueron.gen!J	198
06	Worm:AutoIT	Autorun.K	106
07	Trojan	C2Lop.P	146
08	Trojan	C2Lop.gen!G	200
09	Dialer	Dialplatform.B	177
10	Trojan Downloader	Dontovo.A	162
11	Rogue	Fakerean	381
12	Dialer	Instantaccess	431
13	PWS	Lolyda.AA 1	213
14	PWS	Lolyda.AA 2	184
15	PWS	Lolyda.AA 3	123
16	PWS	Lolyda.AT	159
17	Trojan	Malex.gen!J	136
18	Trojan Downloader	Obfuscator.AD	142
19	Backdoor	Rbot!gen	158
20	Trojan	Skintrim.N	80
21	Trojan Downloader	Swizzor.gen!E	128
22	Trojan Downloader	Swizzor.gen!I	132
23	Worm	VB.AT	408
24	Trojan Downloader	Wintrim.BX	97
25	Worm	Yuner.A	800

Figure 3: Malware families found in the Malimg Dataset

A second use of this algorithm could be implemented into an antivirus and categorize not only the family of the malware but also if the image (executable file) is a malware or not.

C- Assumption of similarity

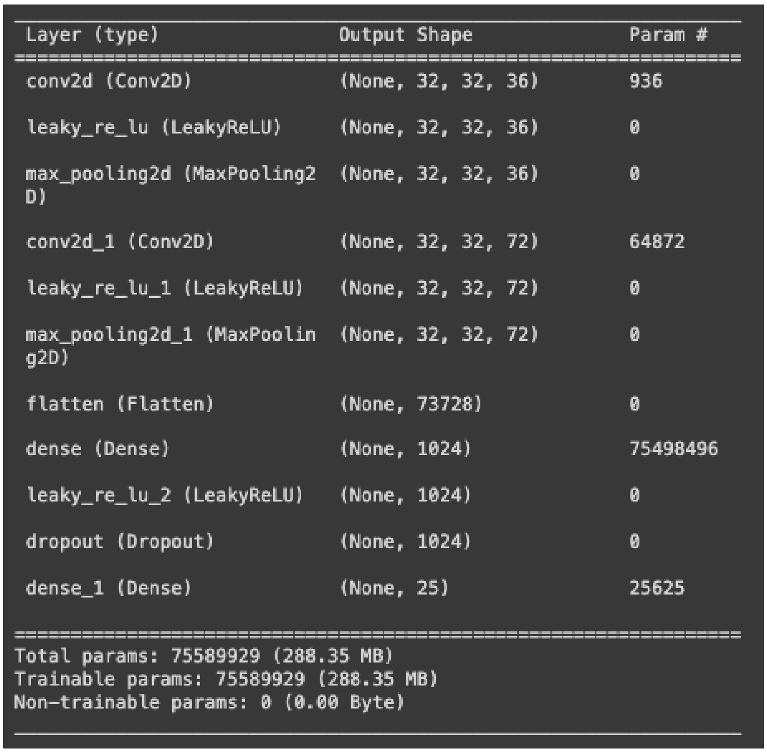
Malware categories exist because various malware have the same way of action or exploit the same vulnerability as worm, trojan or backdoor, etc. We assume that this similarity will also be find in the source code of the malware and likely in the binary file.

So, in our problem of malware classification, looking for virus family likeness amounts to finding common image features. This is why we use convolutional neural network:

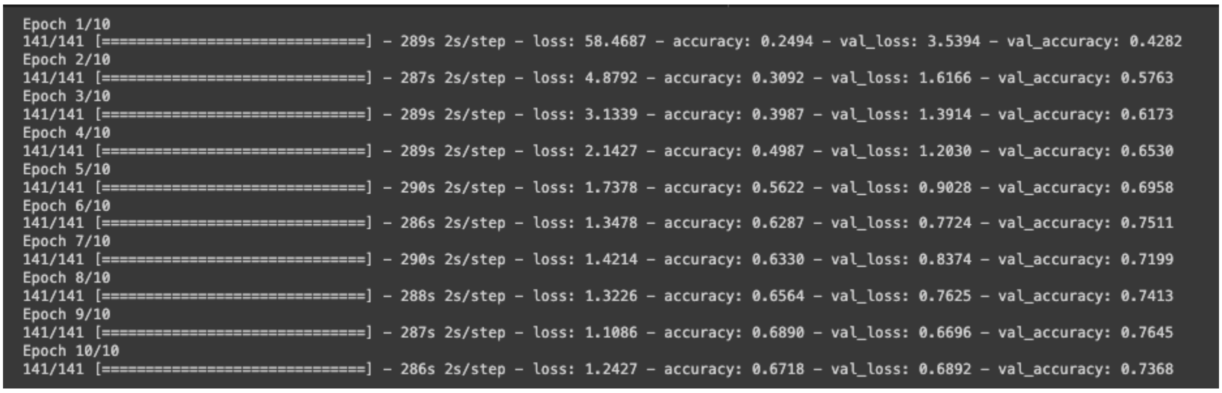
- Exploit the capabilities of CNNs to recognize intricate patterns, relationships, and anomalies.
- CNNs excel at feature extraction from images. They automatically learn hierarchical features at different levels of abstraction, allowing them to capture both local and global patterns.
- Images provide a standardized format that allows for the generalization of features across different types of malwares. CNNs trained on a diverse dataset of malware images can learn to recognize common characteristics shared among various malware strains, enhancing their ability to generalize and maybe detect previously unseen threats.

3- Application of the classification algorithm.

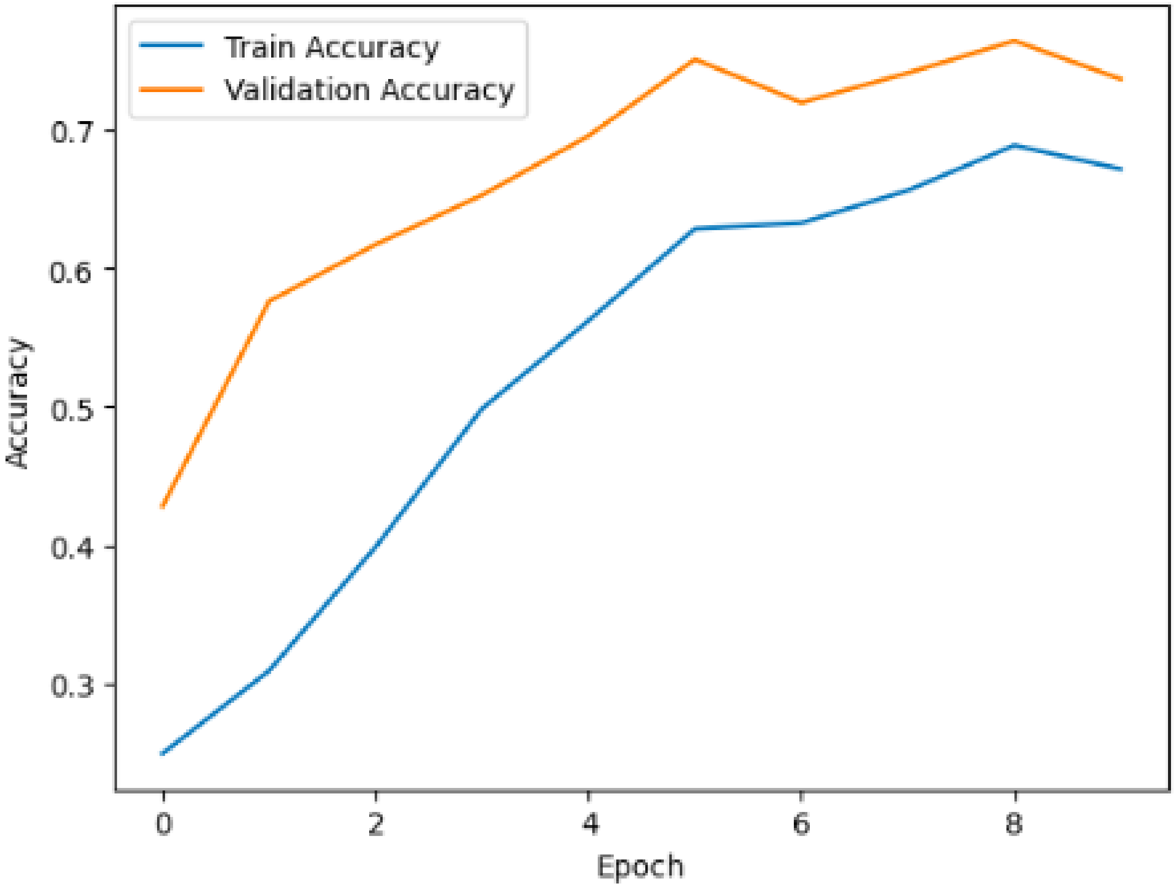
We chose to use the neural network architecture given in the paper :



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 36)        936

 leaky_re_lu (LeakyReLU)     (None, 32, 32, 36)        0

 max_pooling2d (MaxPooling2  (None, 32, 32, 36)        0
 D)

 conv2d_1 (Conv2D)           (None, 32, 32, 72)        64872

 leaky_re_lu_1 (LeakyReLU)   (None, 32, 32, 72)        0

 max_pooling2d_1 (MaxPoolin  (None, 32, 32, 72)        0
 g2D)

 flatten (Flatten)           (None, 73728)             0

 dense (Dense)               (None, 1024)              75498496

 leaky_re_lu_2 (LeakyReLU)   (None, 1024)              0

 dropout (Dropout)           (None, 1024)              0

 dense_1 (Dense)             (None, 25)                25625

=================================================================
Total params: 75589929 (288.35 MB)
Trainable params: 75589929 (288.35 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```



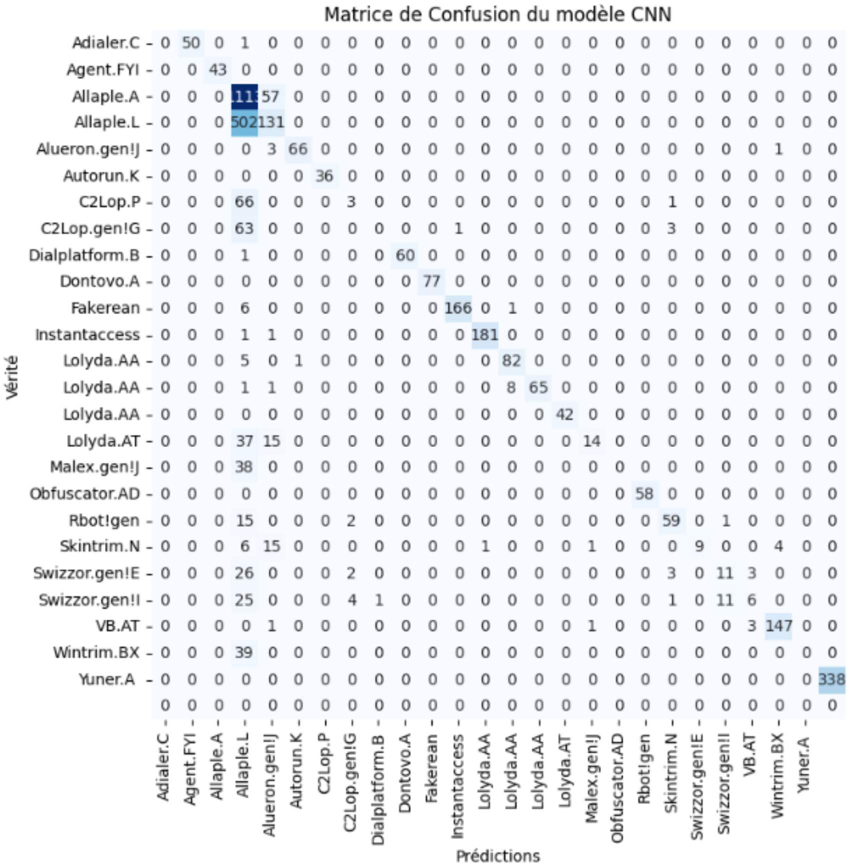
```
Epoch 1/10
141/141 [==============================] - 289s 2s/step - loss: 58.4687 - accuracy: 0.2494 - val_loss: 3.5394 - val_accuracy: 0.4282
Epoch 2/10
141/141 [==============================] - 287s 2s/step - loss: 4.8792 - accuracy: 0.3092 - val_loss: 1.6166 - val_accuracy: 0.5763
Epoch 3/10
141/141 [==============================] - 289s 2s/step - loss: 3.1339 - accuracy: 0.3987 - val_loss: 1.3914 - val_accuracy: 0.6173
Epoch 4/10
141/141 [==============================] - 289s 2s/step - loss: 2.1427 - accuracy: 0.4987 - val_loss: 1.2030 - val_accuracy: 0.6530
Epoch 5/10
141/141 [==============================] - 290s 2s/step - loss: 1.7378 - accuracy: 0.5622 - val_loss: 0.9028 - val_accuracy: 0.6958
Epoch 6/10
141/141 [==============================] - 286s 2s/step - loss: 1.3478 - accuracy: 0.6287 - val_loss: 0.7724 - val_accuracy: 0.7511
Epoch 7/10
141/141 [==============================] - 290s 2s/step - loss: 1.4214 - accuracy: 0.6330 - val_loss: 0.8374 - val_accuracy: 0.7199
Epoch 8/10
141/141 [==============================] - 288s 2s/step - loss: 1.3226 - accuracy: 0.6564 - val_loss: 0.7625 - val_accuracy: 0.7413
Epoch 9/10
141/141 [==============================] - 287s 2s/step - loss: 1.1086 - accuracy: 0.6890 - val_loss: 0.6696 - val_accuracy: 0.7645
Epoch 10/10
141/141 [==============================] - 286s 2s/step - loss: 1.2427 - accuracy: 0.6718 - val_loss: 0.6892 - val_accuracy: 0.7368
```


Train Accuracy
Validation Accuracy
0.7
0.6
0.5
0.4
0.3
Accuracy
0
2
4
6
8
Epoch

A- Confusion matrix analysis

Looking at the confusion matrix, it's like a report card for our anti-malware system. When it correctly spots malware (True Positives), it's doing great. Sometimes, it thinks something is malware when it's not (False Positives), so we'll work on reducing those. When it correctly says something is not malware (True Negatives), that's good too. But, there are times it misses real malware (False Negatives). This tells us we can make our system even better.

As evident in the confusion matrices, the DL-SVM models exhibited higher accuracy in identifying malware families with a significant number of variations, particularly Allaple.A and Allaple.L. This might be attributed to the oversight of the relative distribution of each malware family during the dataset partitioning into training and testing sets.



	Adialer.C	Agent.FYI	Allaple.A	Allaple.L	Alueron.gen!J	Autorun.K	C2Lop.P	C2Lop.gen!G	Dialplatform.B	Dontovo.A	Fakerean	Instantaccess	Lolyda.AA	Lolyda.AA	Lolyda.AA	Lolyda.AT	Malex.gen!J	Obfuscator.AD	Rbot!gen	Skintrim.N	Swizzor.gen!E	Swizzor.gen!I	VB.AT	Wintrim.BX	Yuner.A	
Adialer.C	0	50	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Agent.FYI	0	0	43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Allaple.A	0	0	0	111	57	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Allaple.L	0	0	0	502	131	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Alueron.gen!J	0	0	0	0	3	66	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Autorun.K	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C2Lop.P	0	0	0	66	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
C2Lop.gen!G	0	0	0	63	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	3	0	0	0	0	0	0
Dialplatform.B	0	0	0	1	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Dontovo.A	0	0	0	0	0	0	0	0	0	0	77	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Fakerean	0	0	0	6	0	0	0	0	0	0	0	166	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Instantaccess	0	0	0	1	1	0	0	0	0	0	0	0	181	0	0	0	0	0	0	0	0	0	0	0	0	0
Lolyda.AA	0	0	0	5	0	1	0	0	0	0	0	0	0	82	0	0	0	0	0	0	0	0	0	0	0	0
Lolyda.AA	0	0	0	1	1	0	0	0	0	0	0	0	0	8	65	0	0	0	0	0	0	0	0	0	0	0
Lolyda.AA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42	0	0	0	0	0	0	0	0	0	0
Lolyda.AT	0	0	0	37	15	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0
Malex.gen!J	0	0	0	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Obfuscator.AD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	58	0	0	0	0	0	0	0
Rbot!gen	0	0	0	15	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	59	0	1	0	0	0	0
Skintrim.N	0	0	0	6	15	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	9	0	0	4	0	0
Swizzor.gen!E	0	0	0	26	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	3	0	11	3	0	0	0
Swizzor.gen!I	0	0	0	25	0	0	0	4	1	0	0	0	0	0	0	0	0	0	0	1	0	11	6	0	0	0
VB.AT	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	3	147	0	0
Wintrim.BX	0	0	0	39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Yuner.A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	338
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0