SCIENCES SORBONNE UNIVERSITÉ

Sorbonne University

Master ANDROIDE

Decentralized Test of Liveness

Master 2 Internship

Done by :

Petar Calic

Mentor :
Nicolas Bacca, Ledger (Innovation Lab)
Referal profesor :
Nicolas Maudet, LIP6, Sorbonne University

September 22, 2022

Contents

Chapter 1

Introduction

1.1 Ledger

1.1.1 Company presentation

Ledger is a French company that mainly designs and markets physical crypto-currency wallets. Ledger was created in 2014 by 8 founders with different backgrounds, the technicall one being held by Nicolas Bacca, who then became Ledger's CTO. He is today the only founder left in the company and occupies the position of President of the Innovation Lab. Today there are more than 700 employees working for Ledger, in Paris but also in New York and London. In 2014, Ledger's promise is to secure users' crypto-currencies as well as their transactions when interacting with the blockchain. To do this, the company's engineers are developing physical digital wallets that allow individuals to protect their crypto-assets by giving them exclusif ownerwhip of their own private key. The first successful product was the Ledger Nano S released in 2016. To this day, it is the most popular hardware wallet in the world, having sold more than 3 milion wallets. The french Unicorn has expanded its product range today and is offering services as an app Ledger Live and enterprise solutions: Ledger Vault.

1.1.2 Innovation lab

Research and Developement

During my internship i had the opportunity of being in ledger's innovation team. It's a 6 members team of various skills (one full stack developer with a phd in computer vision, two smart contract developers a phd cryptographer and at Nicolas Bacca at the head of the team. He is an expert in hardware security, monetics and the blockchain) He is the founder of various companies in the field. The team's mission is to keep track of new ideas and projects that could increase's ledger's value and try to keep it one step ahead of its competitors.

Organization of my internship

As i had an AI related project, i was working of the Test of Humanity project alone, but being supervised with weekly meetings with Nicolas Bacca, and being helped by a lot of people of the company. I didn't hesitate to seek help from the Data Science team, the Ledger live team(Full stack developers) and my team when i had specific questions. In the beginning parts of the projects i had multiple sessions with LIP6 researchers as i had some questions about topics that Ledger didn't have experts about. I had to be very autonomous as i was working on my project alone. Aside of my weekly sessions with Nicolas Bacca, i participated in weekly team meetings, conferences and even hackathons, were i could watch and learn from the very best in the industry.

1.2 Subject

Know Your Customer (KYC) procedures are a critical function to assess customer risk and a legal requirement to comply with Anti-Money Laundering (AML) laws. Effective KYC involves knowing a customer's identity, their financial activities and the risk they pose.

Tests of Humanity are a subgroup of tests designated to distinguish real and fake persons (bots). It consists of a set of technical features to counter biometric spoofing attacks where a bot imitates a person's biometrics or characteristics in order to deceive or bypass the identification and authentication steps given by the system. With the emergence of the internet, and with it, the daily increase of our everyday life dependence on its functionalities, increases as well the targets and vulnerabilities of third party malicious intentions. Here is one personal example: I made some digital art images that I wanted to give away freely to people that attended an online event. Two seconds after I shared the link, a bot took all 100 of the tokens. A simple humanity check could prevent that. With AI capabilities to imitate humans rising every day, the task to discover bots that don't wish to be discovered, becomes more and more challenging.

However, there are still very successful tests like Google's CAPTCHA which relies on secret images recognition, mouse movements and a top secret server classifier. Our goal is to make an Test of liveness that is : Open source (No security by obscurity), Decentralized (The test won't be running on a local server but on an autonomous smart contract deployed on a Virtual Machine on the blockchain(STARKNET), that would be accessible to anyone, and more user friendly than the existing tests. There is also a more profound and general mission behind the project which is to make advances in the field of decentralized technologies and use its potential. We should also start relying more on technologies that are verifiable rather than trustable.

An important note is that a test of liveness based on machine learning will never be 100% bot proof as the constant race of technologies between the two sides never stops (Not to mention the existence of human farms for solving CAPTCHAs). In order to be completely sure that we are dealing with a real person the only way is to request a document verification for really critical systems. That means that the development of a 100% bot detection rate algorithm is not the goal of this internship rather than the exploration of new technologies and exploiting their potential.

The objectives and the roadmap of the project are the following:

(a) Take the time to design an architecture of the test that has the following attributes: Able to run on the blockchain (Operational while being open source, doesn't require a lot of memory as it is expensive); Privacy preserving; Is able to run on both phone and computer; Improve User Experience if possible.

(b) Implement a first operational Local version of a Front End and a Back End.

(c) Implement and deploy a decentralized version that runs autonomously.

Chapter 2

Key Notions

As we are presenting with a project that requires a good understanding of the blockchain and how it operates, this chapter is necessary for the reader that isn't familiar with the environment.

2.1 Test of Liveness

A Test of Liveness, also known as Turing Test, is an evaluation of a machine's ability to express intelligent behaviour indistinguishable from that of a human. Originally it is operated by a human. In the case of a machine performing the discrimination process it is called a Reverse Turing Test. A popular example are CAPTCHA's which stands for "Completely Automated Public Turing test to tell Computers and Humans Apart". It is widely used on web sites with the mission to identify human users and block access to users with machine like behaviour also known as "bots". They have a lot of downsides that we are going to tackle in this project: Poor User Experience (UX), a big majority of people think that CAPTCHA's are very tiresome and statistic shows it reduces a lot online traffic on web sites. Nowadays it can take up to 10 seconds do pass CAPTCHA.The tests are becoming more and more difficult as AI bots learn to solve CAPTCHA's. They often rely on the test on being visually a hard exercise, so people with some visually disabilities are discriminated.

Another big problem that we worked on is the fact that for a automatic test to be operational, it needs to store the results of the tests on a local server so that it can judge the correctness of the answer. This introduces two major problems : The single point of failure: If the server or a single element of the architecture is down, all the sites that are using the test don't have an operational test at the moment. Also it relies on secrecy of the answers, if someone leeks or sells information about the test it compromises its purpose.

2.2 Blockchain

The Blockchain is a database. Its a technology for storing and transmitting information without central authority. Technically, it's a replicated storage in which the information sent by users and the links within the database are verified and grouped at regular time intervals into linked blocks, where nodes collectively adhere to a consensus algorithm protocol to add and validate new transaction blocks. Blockchains are secure by design as they can be programmed with a high Byzantine fault tolerance (reliability of transmissions and integrity of interlocutors).

The two most important and popular blockchains are Bitcoin and Ethereum. In the common opinion they are both very similar things, virtual money, but its actually not the case.

Bitcoin is the first successful blockchain technology applied to a use case of a distributed ledger that is keeping track of a virtual token distribution over the network. Bitcoin is also the name of the token (the currency). It is considered the stablest and safest chain.

Ethereum is a blockchain that has also its own currency but has a much wider use case. The Ethereum network also makes it possible to create and run decentralized applications and smart contracts (programs stored on the blockchain that can be run on Ethereum's virtual machine (EVM)). The addresses on Ethereum then are either users, or contract ones. The EVM is the runtime environment for transaction execution in Ethereum. Its instruction set is Turing complete and it includes a stack, memory, program counter, and the persistent storage for all accounts (including contract code). Its essentially a big computer shared by everyone. Each function execution, and even more, state changes (storage modifications) are very expensive. Calling a function of a contract can cost between a few cents to hundreds of dollars depending on network congestion and the type of instructions called. Reading variables is cheaper than writing in storage, but both are considered expensive. That's why scalability solutions are being invented as STARKNET, a zero knowledge cryptography system designed to make computations cheap and maintaining Ethereum's security and stability.

2.3 STARKNET

STARKNET is a Layer 2 Zero Knowledge Rollup created by starkware. Here are more details about it.

2.3.1 Layer 2

Simply put, it is a technology that adds an extra layer to a blockchain to relieve the operation of the base chain. Like Bitcoin, Ethereum can be thought of as a Layer 1 protocol. It's the base layer that stores securely all transactions off the network We have different approaches today of building layer 2 networks as:

(a) Sidechains: Blockchains that run separately from the main blockchain. They operate independently using their own consensus algorithm, laying of the work of the main chain but have the issue of not using the security and decentralization ethereum offers. They're compatible with the Ethereum Virtual Machine, but they're also security limited.

(b) Rollups: They operate by executing transactions on the Layer 2 network, while submitting data to the base chain. This means that they benefit from the security of Ethereum, but can perform transactions outside of Layer 1. This means that writing on storage for example stays expensive as it needs to be sent to L1. There are two types of rollups: ZK rollups based on zero knowledge cryptography, which bundle many transfers into one transaction, and optimistic rollups, which operate in parallel to Ethereum. We will be taking interest in a ZK Rollup: STARKNET.

2.3.2 STARK

A zero-knowledge proof (ZKP) is a cryptographic approach that allows one person (the prover) to verify a claim made by another without the prover reveling crytical information. Zk-stark's are proofs based on Scalable Transparent Argument of Knowledge. The system involves two parties: A prover, and a verifier. The prover generates an integrity proof of a certain computation which can then be verified by anyone, without redoing the computation to validate the execution. The generation of is expensive but the verification is very cheap. Here is a use case for clarification: Machine Learning models are often executed on clouds. How can we be sure that the model was trained as long as we wanted or that the same code was even executed. That's the purpose of stark proofs. In the context of STARKNET instead of generating a proof of the execution trace of a program, we generate it over the state changes (registers) of a Virtual Machine. Then the state changes are pushed to L1. That way computation is very cheap when writing smart contracts on the L2 STARKNET.

2.3.3 CAIRO

Writing programs on STARKNET is particular because of the unusual environment as each machine instruction needs to have a polynomial constraint equivalent to be able to generate ZK-STARK Proofs. This leads to certain constraints that developers are not used to. Contracts need to be written in CAIRO, STARKNET's smart contract language. It's a low-level Turing complete functional language functional. It has specific traits like immutable memory, and not operate on integers but felt's (field elements). Which have unusual proprieties like no decimal representation and unusual behaviors when dividing felts and the result not being a whole number. For example 7/3 returns 1206167596222043737... while the same number multiplied by 3 gives 7 again. So in order to write smart contract that could be deployed on STARKNET, own must learn this language that is not easy to onboard as the project is still in alpha and there are a lot of corrections and not a lot of resources online.

Chapter 3

State of the Art

3.1 Tests of Liveness

The most advanced and reliable test of liveness is Google's CAPTCHA. Which involves users selecting images that correspond to the test's task. It is based on Google's huge and easy access to training data. So essentially they are relying on their supremacy in data access so they can build the most perfect object recognition classifiers. There is probably no other lab in the world with a database of that size. The Test is even self nourishing. Some of the pictures that are presented aren't lablelized, using the test as a means to do that and automatically increase its question Database. Moreover they are constantly adding new Image batches so that no one can gather the picture samples and train over them as the questions change periodically. Google's Test has also another powerful feature: It analyzes the mouse's movement in order to make a more confident discrimination. We couldn't find literature about the exact mechanism of google's CAPTCHA but we gathered information by speaking to LIP6 researcher N. Perrin and Ledger's data team. There exists CAPTCHA v2 which relies solely on mouse movements and V3 on user's web activity. The V2 has the problem of not working on mobiles which today is the main web surfing tool and the other of arguably breaching user's privacy which we want to avoid.

There exist various other interesting methods of detecting bots in. For example this paper describes a new interesting method of detecting bots for mobile phones [9]. It uses a phone's Gyroscope to detect small movements and vibrations when a user interacts with the screen. It has then a pre-trained Machine Learning Model that distinguishes human interaction when solving questions with artificial movements on the phone (staying still, or attached on a pendulum). The drawback of this method is that it works only on devices with a working gyroscope.

To resume the state of the art literature, it is obvious that no method or test could theoretically stop an AI algorithm form learning to pass it for ever. It is a question of time, incentive and budget. It is an ongoing war between bots and designed tests to distinguish them form humans. A problem for a lot of tests is that they augment the difficulty to lover the threshold of bots that can pass the test. That price has to pay the user which now needs more time to pass. Companies producing those tests have to keep being informed always about the advanced in solving CAPTCHA technologies. Using a GAN [3] may counter a lot of tests that analyze human behavior. In general, we need to accept that there exists a constant race between Machine Learning algorithms trying to imitate human behavior and algorithms trying to discriminate between the two.

There exists also CAPTCHA solving illegal sites which pay people for solving tests. Those sites pay poor people 1 dollar per 100-1000 solved CAPTCHA's. They are officially presented as English learning exercises. That's why we theoretically don't even need a 100% performing test of Liveness: Humans can solve the test for other humans. Also because solving those tests is so cheap, an economic study, and predictions about what is protected needs to be made. The v1 CAPTCHA is the most robust one, but because of the previous reason, they made the v2 and v3, which are less robust but more friendly to users. However the risk of a single point of failure exists and keeps growing as the same Test keeps getting used on more and more sites.

In order to make a quality test of Liveness it seemed needed to make research about human perception and its difference of actual state of the art AI algorithms***cite. We found out that even tho performances of today Image recognition algorithms is similar to that one of humans(even better in some case's), it behaves in a different manner in some situations. Which is all we need, not humans outperforming machines in a task, just distinguishable behaviour. In this article[13] we see that machines and humans reacts similarly to Gaussian noise, it appears like bluer for humans. But then if we apply error correction algorithm, that for example makes means of pixels per region, it makes the recognition easier for humans and harder for machines. In this article [12] they developed a particular noise that is unnoticeable to the human eye but makes machine learning algorithms fail on databases they were trained on.

3.2 On the blockchain

By doing a lot of research we found that there is still no work done in the decentralization aspect of such an application. There are a few reasons that we assume might cause this. There are a few problems that the fact that the code needs to run on the blokchain are making the development of such an application difficult: Expensive computation and the fact that the blockchain is public. So even tho variables and functions in a contract can be made public, it is not very hard to reverse engineer the byte code stored on the blockchain.

There is a protocol on Ethereum called Proof of Humanity. It does the task of task of discerning humans from bots but it's architecture isn't satisfying our goals. Firstly it works by social verification, another human needs to attest you are really a human by being himself verified. That is hardly a scalable solution and seems unsafe to prove forever that a certain address on the blockchain is detained by a real person. As mistakes can lead to more bots enter the verified circle. A malicious user could also verify bots even if the protocol doesn't incentivies that. The second problem is that it asks the user to turn on his camera which we want to avoid. For Privacy reasons, user experience reasons, and for safety reasons. Recent advances in GAN generated faces shows that it is no longer safe to assume that if a picture or video of a human exists it exists in real life.

With the high probablity of need for this kind of project to use Machine Learning models we searched for research done in the blockchain area. Very little work exists because of the expensive computation limitation. But with the possibilities STARKNET offers, computation is now cheap so all we have to worry about is memory. A Blockchain researcher showed it is possible to run Multi Layer Perceptron Inference***cite on chain. He built a small model of two fully connected layers that he trained locally using Pytorch on the mnist database, and then exported the weight manually and rebuilt the network architecture using cairo into smart contracts. This way its possible to call an inference contract with the model that lives on chain. However it is important to note that even for this small architecture it is very complicated to program the model in Cairo. A first reason is that as storage is still expensive we can't save variables into storage for later use. We have to make a script that will print out cairo code with constants values in it in order to perform each operation of the sum of product. The second limitation is that for now that STARKNET's prover parser has a limit of the number of operations a contract can contain as well as a constraint on the size of the contract. Today the limit is around 300 sums, multiplication per function and 40MB of code per contract. Because of that we have to make a script that will split the contract into more contracts which will call one another. For the multi layer perceptron with a model of around 25000 parameters, they needed to deploy 300 linked smart contracts. Link of the code in Appendix.

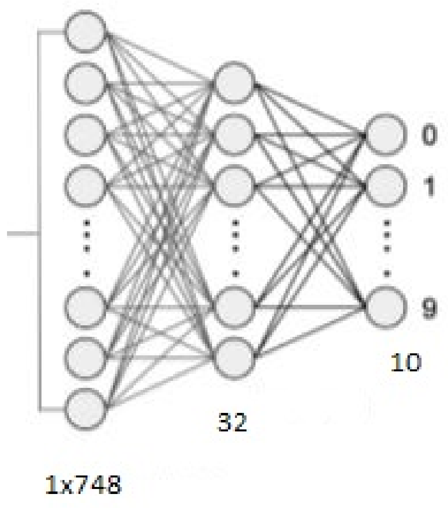


Figure 3.1: Architecture used in the contract

Chapter 4

Contributions

4.1 Architecture of the Test

Being aware of the size and number of steps of the project, we understood that there was no room for big corrections. We had to do good in depth research and conceive a good architecture from start to finish before starting to code.

We had to keep in mind always the global goals of the project:

- 100% Decentralized test
- privacy preserving
- Good User Experience (Human Machine interaction)
- Works on all devices

The most important point is the first one. We focus primarily on succeeding to accomplish that point first and then managed to add other points as much as possible.

4.1.1 Decentralization

In order to succeed in making a test of liveness completely without a server meant to find solutions to several issues:

- expensive computation.
- possibly having to do machine learning inference on chain.
- The blockchain is public, so we can't store critical information in the contract. The test can't store the solutions to the test.

The first point is already solved if we build the test not directly on L1 but on STARKNET's zk-rollup. The only draw back is that we will be building on a project that is still very new and in alpha can lead to unexpected issued and bugs that are hard to solve.

Machine Learning on chain is also solved thanks to STARKNET as we seen in the ***cite contract's example. The drawback is that we are limited to very small models as even a small MLP requires complicated architecture.

Nevertheless those drawbacks are manageable, the key part of the project is to find a solution to that of public storage which means we can't store. An interesting thing to note is that ZK methodology that STARKNET is based on focuses on scalabilty for now. But it is expected in the future that they tackle also privacy. For now we will have to find a solution to this classic blockchain problem.

In order to make it hard for an AI to solve the test we want to combine visual challenges inspired by Google's CAPTCHA V1 (as it satisfies other points and demands the bot some AI skills, which is not trivial to implement, and human behaviour metrics.

We have seen that [3] mouse movement, gyroscope sensor [9]... Work well for this use case, but if the Machine Learning model is public, we can't rely solely on it because an attacker could use the model to train a GAN that generates mouse movements for example. So we need to use it in combination with a challenge that's hard for an AI to solve in order to make the distinction with human behaviour more obvious. How to check if the user passed the challenge without storing the answer to the test or doing the test. The answer we came up with is to not make the taste identical on the side of the user and the verifier.

Consider the following scenario we have designed for our test:

See Appendix A. figure 2

The server generates a big image with numbers and letters. Now the user has to localise the prompted number and click on it. A small local screen shot is made of the zone he clicked on and sent to the server. Now the server (the verifier) has only the challenge to say if it was the correct answer which is easy even for a small ML model. This way the task is harder on the user's side which is the logic we want were looking for. We can then add random perturbations when generating the image like we saw in articles [13] Of course as in the decentralized version, the code is in a smart contract and not on a local server, one can learn a model to learn over the precise perturbations so it adds only a few steps more for an attacker.

4.1.2 Privacy

We discussed using the camera for face detection, or ocular movements in order to add complexity to he test to be passed, but after a discussion with the legal team at Ledger, we understood that making those types of maneuvers requires complicated and precise implementation pipelines that need to be respected and authorized, which isn't what we are looking for in an innovation project.

4.1.3 Good User Experience (UX)

On the DevConnect conference in Amsterdam in april 2022, it was said about human perception that a waiting time of less than one second on loading is considered a bug and has little to no influence on the UX. between 1 and 10 seconds the user considered there is a problem, affecting UX but the user doesn't leave the page. After 10s the user considers the page broken and is likely to leave. We decided to apply that threshold, so we want to make a test that takes less than 10 seconds to pass. Also we want to test to be visually very simple so the user isn't faced with visualy stimulating or new layout every time he does a test. We have to keep the design as clean as possible. That's why the design will be to put the test in the center of the screen in a neutral blue color window. The question type of question will be the same every time: Find the number X. That way the user doesn't need to read the task every time. The test is robust enough when combined with human behaviour metrics. Also we have to keep in mind the existence of human CAPTCHA solving farms that are pretty cheap, so pushing the limit on the side of the user isn't the approach to choose we believe.

We measured google's CAPTCHA test that showes up on discord, and found a mean time of 14seconds on 20 tests. The measures where done on two 23year old persons with long brake intervals to minimize the learning effect on the specific test. See picture in Appendix: Figure A.1.

4.1.4 Device compatibility

With good front end responsive implementation the test will have a similar layout on all machines. An important note is that in the collection of the learning data for human behaviour characteristics they will have to be learned on various devices independently. On computer the test should use mouse trajectory, and on mobile the gyroscope. As this is proven to work well, it will not be implemented in this project but is a key element to a functional test

4.2 Centralized Test of Liveness

Due to the complexity of the project it was advises by the team to first implement a functional version of the test on a local server. See code in Link 2 in Appendix B.

4.2.1 Front End

In order to keep the web app of the test as portable as possible it was advised by the ledger front end developer team that front end is programmed purely in Java Script (In combination with HTML and CSS) which will also benefit the simple layout of the page. Also Front end developing libraries and frameworks have an important learning curve which is not the subject of the internship.

The Front End needs to do the following:

- Fetch the JSON file from the server containing the whole picture to display.
- Extract Array with pixel values
- Render using Canvas
- Implement Screenshot logic that extracts the corresponding clicked small array of pixels
- Pack to JSON and send back to server

4.2.2 Back End

It is important to note that the Front End is considered untrusted when developing web applications. For example the generation of the picture cold be easily tampered with, so it needs to be done on a local back end.Front end is used only as a Client for displaying information and capturing inputs from the user. The App logic needs to be implemented on a server.

The role of the server here which we implemented in Python using Flask framework:

- Fetch random MNIST number pictures from the DataBase. Glue them together randomly so that an attacker won't be able to predict the localisation of each number.
- Pack Array to JSON with the random question to ask and prepare to respond to command "GET" HTTP request.
- When the server receives a "POST" request it unpacks the snapshot sent by the front.
- Check if corresponds to the question with ML model trained to recognise digits
- Send to front if the test was passed or not

4.2.3 Machine Learning Model

In order to verify the response we had to train Machine Learning model to recognize the snapshots sent to the server. We used Pytorch to train a Convolutional Neural Network. We ran comparisons of performances on different models but the only acceptable one was the CNN since the Multi Layer Perceptron even with a lot of hidden layers is not able to learn on digits that change positions in the snapshot. Which is the case in our example since the snapshot the user creates is of a position that's completely arbitrary.

4.3 Decentralization

We arrive at the core of the project which is implementing CNN on chain inference. The difficulty is double: Since there are no existing libraries, we will have to immplement the CNN architecture from scratch by hand. So a very good theoretic understanding of the model is necessary. The second difficulty lies in the unusual programming logic we have to use when programming Cairo STARKNET contracts and the absence of useful libraries like numpy in python. To represent matrices for example we operate on vectors that are flattened and to do matrix multiplication we need to implement recursive functions with basic sum product operations.

The first step was to implement the CNN structure unused in the centralized version by hand in a low level mindset in order to grasp well the whole architecture.

Then in order to simplify as much as possible the Cairo implementation, we chose a simpler architecture that gives also good results (95% of classification): One single convolutional layer containing 7 filters.

When implementing the contracts in Cairo we have to keep in mind the limitations of the number of operations per contract do we have to divide the contract as follow:

- Master contract, sending the input data vector of length 784 all the layers of the model and returning the result.

- It holds addresses of CL-feature<0-6> (7 other contracts each responsible to return one of the 7 Feature Map Layer) so he will be able to execute them n his own. Each CL-feature contract disposes of one filter of size 5x5 stored as a vector of constants that it uses to do the convectional operation over the picture input.

We got information from the Starkware team that the number of sums and multiplication per function must have been raised so we will have to wait for deployment of the contract to see if it works. In case not simply divide further the contract this way:

- Each CL-feature contract calls 24 (width of the feature map) times a CL-feature-subop<0-23> contract which will perform himself 24 convolution operation with the filter over the input image.
- We implemented the recursive Conv-Operation and snapShot function which extracts from the data vector of size 784 (28x28) ,a window vector of size 25(5x5) necessary for the convolution operation at each step.

The master contract then flattens the 7 feature maps returned by the Convolutional Layer and sends it in the same way, thru the following contracts: MAXPOOL -> Relu -> FC(fully connected layer). Architecture is identical to this contract except with less parameters so less contract division. The master contract returns finally a vector of size 10 and the argmax function will correspond to the model's prediction.

As once deployed smart contract code can't be modified, the testing step of development is critical. For security reasons and to be sure that the implementation is final. We used (PROTOSTAR, a python compilation and testing framework for Cairo contracts) to do unit tests of our cairo functions.

Due to lack of time we didn't manage to write a python script that will deploy all the contracts and link them together so they can continue living on the blockchain. However the most important part of the project was implementing and testing the Convolutional Layer since the fully connected layer has already been implemented.

Chapter 5

Results

5.1 Security reviews

During the whole internship, other then the discussions i had with LIP6 researchers and Ledger experts, Nicolas Bacca organised after every important cycle, a session with Jean-Baptiste Bedrune, Ledger's Head of Security.

The first reunion was 1 month ad a half after the start of the internship and had the role to give me feedback about the project's directions and research done. His advises and remarks helped reshape the architecture of the project and gave new ideas about research. The main remark was against using any closed systems as protection and the dangers of open source programs that are used as security apps.

The second reunion was the most important one, as we had the centralized version of the test finished and he could give input about the corrections needed before passing to the decentralized version. He had remarks about two main security flaws. The architecture of the test can't be stateless. The advice of a Ledger engineer to make the back end stateless(no sessions recorded when a user tries to answer the test, everything happens in real time), the architecture is much lighter but the server doesn't keep track of the link between the chosen question and user's response. For example an attacker can prepare an image of a number 5 and wait for the question to be : Find the number 5. He sends that picture to the server without even clicking on the test picture. We should change the architecture to be statefull, and keep a Database of the current user, and the question sent, while giving him some time to give a response back. The second problem is that even the tho the server can keep track of the number of tries, a user can download the image and run ML object recognition over the image ***cite and post 1 answer. The human behaviour metric like mouse movements, time of response could help counter that but the only way to verify that would be to finish the project and send the Test to Ledger's security team (The Donjon) so they can try an attack the system and give back a proper evaluation of the safety. As our focus is on decentralized systems and innovations in that environment, we continued the project with the original architecture.

5.2 Validation tests

This was brief part of the project where we wanted to validate our UX design choices. We compared our test with google's CAPTCHA and expected for the experience to be at least as good as the test is interaction wise a lot simpler.

The experiment consisted of simply asking 14 candidates which test they preferred and why. The first question was asked to 7 people after comparing google's CAPTCHA to one of resolution of our test. All candidates confirmed they preferred our test of Liveness as it was faster and simpler to solve. Remarks were made on small buttons.

The other 7 compared to google's CAPTCHA compared to 3 test resolutions. What we were checking is would the results be the same stay the same if we asked to do the question repeated times in case the test isn't robust enough. Since that happens with google's test also. The responses were mixed, 4 preferred to answer to google's test once than 3 times ours. Since the population is really small we couldn't conclude anything.

Tests limitations: Other than the population of tested people being really small for a serious study it should be a lot more varied. In terms of categories of age and frequency of usage of web services as it may affect the experience.

5.3 CNN vs MLP

While we were choosing the Machine Learning architecture that was the simplest to implement but still had good results, we concluded two important things buy running some tests in Pytorch: The first is is that simple fully connected layers are not adapted for our test of liveness as they don't learn well on objects that change positions. That's due to the fact that the neural network in this case learns on raw data. (a vector of activated pixels of the number 1 are completely different from an image of a 1 translated to the left)

Tranlsation during training	Translattion during test	Classification on test sample
0	0.3	22%
0.2	0	96%
0.1	0.3	40%
0.2	0.3	70%
0.2	0.2	83%

Table 5.1: MLP classification results in function of the translation amount

Notes: The translation coefficient indicator how much the digits were the space of moving in a certain direction rather than being in the center. 0.1 corresponds to small moves, and 0.3 could translate the digit to the border of the image. We conclude that we must use a different class of algorithms: CNN which learns features of an image and is thus immune to translations.

The most important thing we learned about this project we learned is that CNN are perfectly adapted for STARKNET and thus can easily outperform today's state of the art Machine Learning algorithms that are operating on chain. This has to do with the fact that thanks to STARKNET computation can be considered cheap. CNN have the huge advantage over classic NN that they are way more powerful and require to learn way less parameters.

Conv. layers	filters	filter size	fully connected layers	epochs	classif.	parameters
2	8	5	1 fc 128 neurons	7	95%	3106
2	6	5	1 fc 96 neurons	14	94%	2032
1	6	5	1 fc 864 neurons	14	93%	8806
1	6	5	2 fc 864+50 neurons	10	95%	43916
1	7	5	1 fc 1008 neurons	20	94%	10272
2	5	5	1 fc 80 neurons	14	93%	1570
2	6	7	1 fc 24 neurons	13	91%	2320

Table 5.2: CNN architecture and number of parameters

Important: those are the results of training and testing on batches with a translation coefficient of 0.2. We got 98% classification with more filters but in order to keep it simple we decreased their number. As we can see the best performing architecture has 2 convolutional layers with 8 filters each of size 5x5. (Each layer is of course followed by a maxpool layer and relu activation function.). This architecture needs then a Neural layer of 128 neurons. It manages to have 95% of classification performance on the test batch by learning only 3106 parameters. It's important to note that the number of neurons in the fully connected layer can't be chosen. It depends on the number on the convolutions layer parameters.

The precedent experiment with the simple neural network of 2 fully connected layers managed to get on the same test only 83% of correct classification having to learn 25000 parameters. With CNN's we manage to get better results with 10 times less parameters. In the context of the blockchain were memory is very expensive, this type of architecture seems a lot more appropriate. Even tho it seems more costly in terms of number of computations, on STARKNET it will be in fact a lot lighter. Machine Learning architectures that grow in "width" as CNN, by adding more convolutional and maxpool layers, gain performance by not using a lot of weights but rather "recycle" the computation by sending the current layer's output in the next layer's input.

Chapter 6

Conclusion

We have seen during this project how a seemingly simple application like a test of liveness, can become very complex when it serves as a security element. As the race between bots impersonating humans and bot detecting algorithms we have to keep looking for more creative ways of doing reverse Turing tests by thinking about all the trade off. By exploring STARKNET and its possibilities we have showed that it is possible to run even more complex Machine Learning algorithms on chain for inference and that as technology advances and STARKNET matures, possibilities will be even more interesting. CNN are definitely more adapted for the case of starknet as computations are cheap and storage expensive. This will become even more important when we will start to do on chain learning, because network parameters will have to be stored in storage probably which is the most expensive operation.

The size and specter of knowledge required for such a project being considerable we didn't manage to make it to the end, however the project will become public for people alongside me, who wish to continue contributing. My lack of experience in web technology and knowledge on security topics made the first part of the project involving brainstorming, research and tests part of the project very long.

I learned a lot during the internship and i got to see and acquire habits engineers have in a R&D team in a hi-tech company. Also i could participate in weekly team activities which made me feel part of the team. What needs to be done next is to write a script that will deploy the CNN contracts on chain since new bugs can emerge as the ones related to the size of a contract or operations in one function.

In order to have a full decentralized Test of Liveness the next steps would be to implement the rest of the logic in Cairo Smart Contracts(The same one we did in python). Then to have a full robust test human behaviour metrics must be used in the discrimination process like mouse trajectory, time of responses etc...

Bibliography

[1] Vinit Kumar Gunjan, Sabrina Senatore, Amit Kumar, Xiao-Zhi Gao, and Suresh Merugu, eds. *Advances in Cybernetics, Cognition, and Machine Learning for Communication Technologies*. en. Vol. 643. Lecture Notes in Electrical Engineering. Singapore: Springer Singapore, 2020. ISBN: 9789811531248 9789811531255. DOI: 10.1007/978-981-15-3125-5. URL: http://link.springer.com/10.1007/978-981-15-3125-5 (visited on 08/31/2022).

[2] Meng Joo Er, Shiqian Wu, Juwei Lu, and Hock Lye Toh. "Face recognition with radial basis function (RBF) neural networks". en. In: *IEEE Transactions on Neural Networks* 13.3 (May 2002), pp. 697–710. ISSN: 1045-9227. DOI: 10.1109/TNN.2002.1000134. URL: http://ieeexplore.ieee.org/document/1000134/ (visited on 09/02/2022).

[3] Alejandro Acien, Aythami Morales, Julian Fierrez, and Ruben Vera-Rodriguez. "BeCAPTCHA-Mouse: Synthetic mouse trajectories and improved bot detection". en. In: *Pattern Recognition* 127 (July 2022), p. 108643. ISSN: 00313203. DOI: 10.1016/j.patcog.2022.108643. URL: https://linkinghub.elsevier.com/retrieve/pii/S0031320322001248 (visited on 09/02/2022) (pages 7, 11).

[4] Antreas Dionysiou and Elias Athanasopoulos. "SoK: Machine vs. machine – A systematic classification of automated machine learning-based CAPTCHA solvers". en. In: *Computers & Security* 97 (Oct. 2020), p. 101947. ISSN: 01674048. DOI: 10.1016/j.cose.2020.101947. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167404820302236 (visited on 09/02/2022).

[5] Fatmah H. Alqahtani and Fawaz A. Alsulaiman. "Is image-based CAPTCHA secure against attacks based on machine learning? An experimental study". en. In: *Computers & Security* 88 (Jan. 2020), p. 101635. ISSN: 01674048. DOI: 10.1016/j.cose.2019.101635. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167404818312185 (visited on 09/02/2022).

[6] Dorjan Hitaj, Briland Hitaj, Sushil Jajodia, and Luigi V. Mancini. "Capture the Bot: Using Adversarial Examples to Improve CAPTCHA Robustness to Bot Attacks". In: *IEEE Intelligent Systems* 36.5 (Sept. 2021), pp. 104–112. ISSN: 1541-1672, 1941-1294. DOI: 10.1109/MIS.2020.3036156. URL: https://ieeexplore.ieee.org/document/9257172/ (visited on 09/02/2022).

[7] Shibasis Patel, Anisha Sahoo, Bhabendu Kumar Mohanta, Soumyashree S Panda, and Debasish Jena. "DAuth: A Decentralized Web Authentication System using Ethereum based Blockchain". In: *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*. Vellore, India: IEEE, Mar. 2019, pp. 1–5. ISBN: 9781538693537. DOI: 10.1109/ViTECoN.2019.8899393. URL: https://ieeexplore.ieee.org/document/8899393/ (visited on 09/02/2022).

[8] Smita Athanere and Ramesh Thakur. “Blockchain based hierarchical semi-decentralized approach using IPFS for secure and efficient data sharing”. en. In: *Journal of King Saud University - Computer and Information Sciences* 34.4 (Apr. 2022), pp. 1523–1534. ISSN: 13191578. DOI: 10.1016/j.jksuci.2022.01.019. URL: https://linkinghub.elsevier.com/retrieve/pii/S1319157822000325 (visited on 09/02/2022).

[9] Iñigo Querejeta-Azurmendi, Panagiotis Papadopoulos, Matteo Varvello, Antonio Nappa, Jiexin Zhang, and Benjamin Livshits. “ZKSENSE: A Friction-less Privacy-Preserving Human Attestation Mechanism for Mobile Devices”. en. In: *Proceedings on Privacy Enhancing Technologies* 2021.4 (Oct. 2021), pp. 6–29. ISSN: 2299-0984. DOI: 10.2478/popets-2021-0058. URL: https://petsymposium.org/popets/2021/popets-2021-0058.php (visited on 09/02/2022) (pages 7, 11).

[10] Hongwen Kang, Kuansan Wang, David Soukal, Fritz Behr, and Zijian Zheng. “Large-scale bot detection for search engines”. en. In: *Proceedings of the 19th international conference on World wide web - WWW ’10*. Raleigh, North Carolina, USA: ACM Press, 2010, p. 501. ISBN: 9781605587998. DOI: 10.1145/1772690.1772742. URL: http://portal.acm.org/citation.cfm?doid=1772690.1772742 (visited on 09/02/2022).

[11] Darko Brodić and Alessia Amelio. “Analysis of the Human-Computer Interaction on the Example of Image-Based CAPTCHA by Association Rule Mining”. In: *Symbiotic Interaction*. Ed. by Luciano Gamberini, Anna Spagnolli, Giulio Jacucci, Benjamin Blankertz, and Jonathan Freeman. Vol. 9961. Cham: Springer International Publishing, 2017, pp. 38–51. ISBN: 9783319577524 9783319577531. DOI: 10.1007/978-3-319-57753-1_4. URL: http://link.springer.com/10.1007/978-3-319-57753-1_4 (visited on 09/02/2022).

[12] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. arXiv:1412.6572 [cs, stat]. Mar. 2015. DOI: 10.48550/arXiv.1412.6572. URL: http://arxiv.org/abs/1412.6572 (visited on 09/02/2022) (page 8).

[13] S Tiago. “Deep Convolutional Neural Networks and Noisy Images.” In: 2017. URL: https://sites.icmc.usp.br/moacir/papers/Nazare_CIARP2017_DNN-Noise.pdf (pages 8, 11).

[14] Drgon Matus. “Robust KYC via Distributed Ledger Technology.” In: 2020. URL: https://www.researchgate.net/profile/Matus-Drgon-2/publication/346485525_Robust_KYC_via_Distributed_Ledger_Technology/links/5fc4abb8a6fdcc6cc684f4
Robust-KYC-via-Distributed-Ledger-Technology.pdf.

Appendix A

Pictures

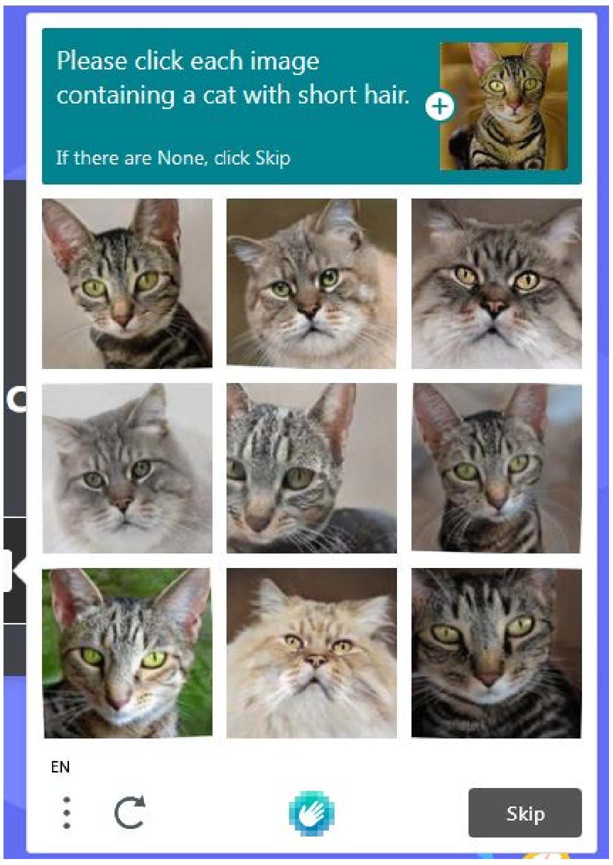
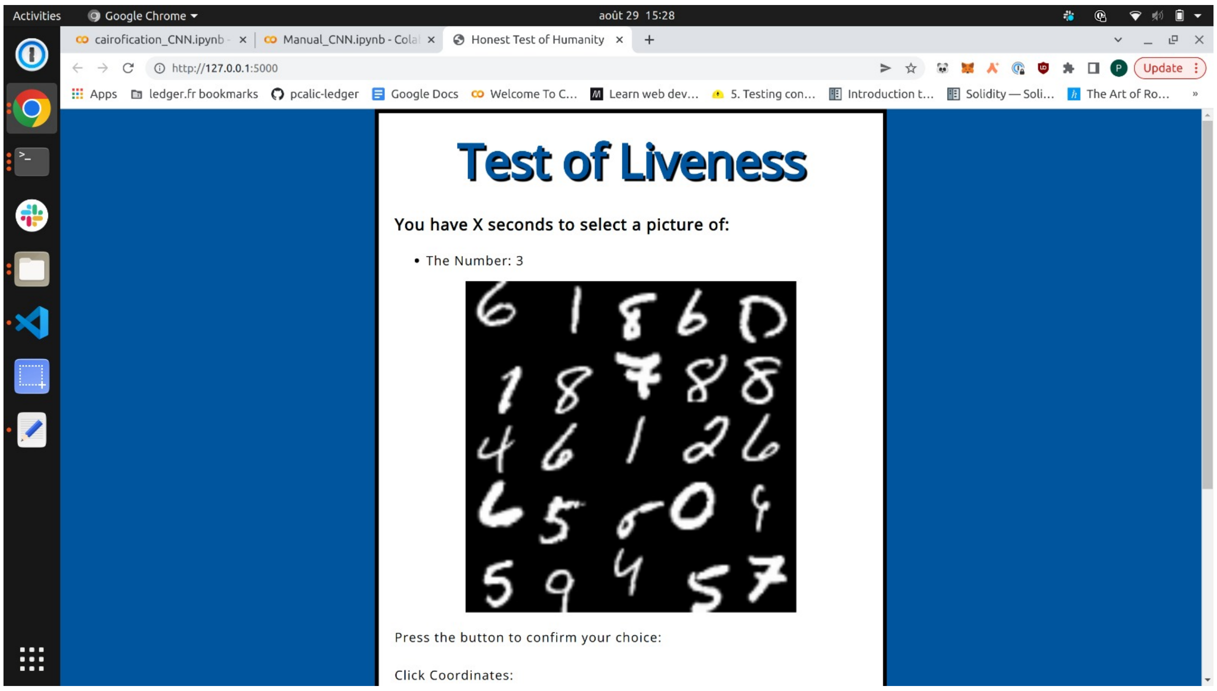

Please click each image containing a cat with short hair.
If there are None, click Skip
EN
Skip

Figure A.1: Google's CAPTCHA



Test of Liveness

You have X seconds to select a picture of:

- The Number: 3

Press the button to confirm your choice:

Click Coordinates:

Figure A.2: Our Test of Liveness

Appendix B

Links

First Machine Learning inference on chain

Repository of the centralized Test Of Liveness

Repository with all the notebook experiments