Securely relying on the crowd

Abstract

As the world becomes faster paced and we generate unprecedented amounts of data, we need to find novel ways to process it on time for it to still be useful. Artificial intelligence is doing an amazing job of getting us closer, but there are always a considerable amount of one-offs, as well as data classification and model training work that needs to be done. For those, the concept of “the crowd” is becoming popular; thousands of persons ready to do small enough tasks for a few cents can help us process our big datasets in a matter of minutes.

This approach to data processing introduces new privacy and data security concerns, such as unskilled or fraudulent workers, bots, data exposure or unintentional data leaks. Here I will share our approach to tackling these risks by correctly placing different technologies to help workers perform said tasks, allowing us to put more controls into place and keep a closer eye on what is being done to the smallest of detail.

Background

As Artificial Intelligence (or, AI for short) advances, we become less tolerant of getting slow results. What used to be a task that was expected to take a few weeks, we want to see solved in days, what used to take days, we want in hours or minutes. But AI, as any technology, isn’t perfect. There will always be tasks that it won’t be able to do, be it one-offs, corner cases or tasks that are simply not suited for it yet-- there will always be the need for a human brain.

Since we humans aren’t able to go through large quantities of tasks as fast as machines can, we need to compensate by having multiple people do small chunks in order to get results in a timely manner, but the operational cost of having large groups of people to handle small tasks that machines can’t do is something most companies do not want to invest in.

To fill the gap, services like Mechanical Turk[1] and CrowdFlower[2] have emerged, offering an API for you to push small tasks for humans to pick and solve. Tasks like filling out 1000 polls, identifying animals on a repository of pictures, or extracting data from truckload of images can be performed within an hour by a few of the thousands of workers signed into the platform. Even

[1] https://www.mturk.com/
[2] https://www.crowdflower.com/

platforms like HackerOne[3], BugCrowd[4], Uber[5] or TaskRabbit[6] rely on more verticalized crowds in similar fashion to power Bug Bounty programs, car rides or home tasks.

These applications of the crowd post both privacy and operational concerns. For that we need a strong framework to distribute tasks to humans, from filtering out bots and unskilled workers, to continually assessing the work being done.

Identified Risk

First we need to be aware of the risk we will be dealing with. Some of them might be obvious, some of them not so much, but both with terrible consequences for our operations.

Undesired Workers

Probably the most obvious concern when working with the crowd is how to make sure the person receiving a task I require done can be trusted or not. Crowds are full of fully skilled and trustworthy people as well as unskilled, malicious, fraudulent or even automated bots workers.

With unskilled workers we are going to get poorly done tasks, which is going to affect our operations greatly, as we won't be able to make use of said tasks' outcome. We might as well spend more time re-submitting the task for another worker to do, or even fixing the bad job that was done. Depending on how customer-facing the results are, we might see our reputation affected as well.

In the same way, fraudulent workers' main purpose is to maximize pay while minimizing time and effort. This type of worker won't follow instructions or even try to perform the tasks at hand, they will just click buttons until they get to the end of the task, so they can collect the pay at the end. Same as with unskilled workers, the outcome of their tasks isn't something we can use.

With malicious workers, our risk varies depending of the goal of said worker. They might intend to steal data, or damage our reputation by entering offensive language on text inputs we present to our customers. These workers are the closest thing we will have to insider threats.

Finally the bots main purpose is highly probably to collect data we expose to the crowd. They will still take our tasks, not letting a real worker take it, and we won't see any of the desired outcome, as the bot is not even going to try to perform the task.

---

[3] https://www.hackerone.com/
[4] https://www.bugcrowd.com/
[5] https://www.uber.com/
[6] https://www.taskrabbit.com/

Data Leakage

In order for workers to perform our tasks we might have to trust them with some PII (Personally Identifiable Information), PHI (Personal Health information), or just sensitive information from our ourselves or our customers. Some of examples of this might be phone number and address in order for the driver to pick a customer up, in the case of Uber, or credentials to our customer's production site for a bug bounty program. Of course, each person who has access to the data is a potential for data leakage, so there isn't much new on that front. However, some of the leakages come from unexpected sources.

One common way of data being leaked comes from workers having questions on our tasks. It may be that they aren't unsure of why their task was rejected or why they got a bad rating.They may not fully understand instructions, or have found a particular corner case and are unsure how to proceed. The usual solution is to look for help, in a honest attempt to perform the task in the best way. The problem with it, is that it might involve taking screenshots of the task, including private data, and posting online in forums or public web pages where workers meet.

Worker Intake

Worker intake will become one of the pillars of our engagement and our first line of defense. With a correctly defined worker intake process, we can eliminate unqualified workers, and even bad actors or bots, without much hassle. This of course can't be perfect, and some of those could still go through, but for anyone who does, other mitigations will be taken later in the engagement.

Requirements and Qualifications

When creating a project on our crowd provider we are given the option to set up Requirements and Qualifications[7] filters on our workers. This will allow us to rely on the worker's history and their score on the provider (qualified by other Requesters) to define their trust level and other characteristics like location. This is especially important for privacy purposes, because it rules out most maliciousl accounts or bots and leaves us for the most part with hard working, honest humans. It's also possible for people to sell their accounts on forums, so we can't solely rely on this feature, but having it as a first filter will take us a long way.

[7] For example in Mechanical Turk we can make use of:
https://blog.mturk.com/tutorial-understanding-requirements-and-qualifications-99a26069fba2

Work Posting

Another great feature we can make use of at work creation time is what Mechanical Turk refers to as External Question[8]. This will insert an iframe on a page we provide, passing some information as GET parameters so we can identify which worker is doing the work and for which posting. This will allow us to have full control of how our data is shown to the worker, and, since we control the page rendered in the iframe, we can enable as many controls as we need to avoid data being mistreated. For instance, this could be a separate set of credentials on our side tied to their MTurk profile with forced Multi-Factor Authentication.

Worker Training and Assessment

Since the crowd provider filters aren't enough and probably don't cover our specific tasks, we will need to provide training and assessment for those on our own. This will help us ensure privacy in two very important ways: initially it will reduce the amount of malicious or lazy workers, and rule out bots or automated accounts; in the long run it will ensure that workers are familiar with the rules and guidelines to follow and have some familiarity with most of the cases they are going to encounter. This will reduce the probability of workers posting questions, potentially containing screenshots with private data, on forums on an honest search for answers as they are unsure of how to proceed.

Of course this isn't a one-time thing, and we need to keep assessing as the worker keeps coming for more assignments. This might take the form of some random tasks for which we know the expected result, so we can correctly assess how well the task has been done. One thing to take into consideration is that automated tools as well as muscle memory play against us here, and if the workers keeps getting the same task over and over again, they will learn to answer without even doing the task, as the answer is the same as last time. We should find a way to send the same task with small variations here and there, so that every time the same task is assigned, a different result might be the correct one.

Work Confinement

Since we are using our custom application for the workers to perform the tasks, we can be as careful as we want to be. We can confine the data processing and handling to a specific, isolated and highly-controlled instance. With current container and Virtual Machines technology we could provide a virtual machine for the tester to use, to which the tester will connect using our web application (through a web based VNC client, for example). Since we control the stack

---

8 https://docs.aws.amazon.com/AWSMechTurk/latest/AWSMturkAPI/ApiReference_ExternalQuestionArticle.html

of virtual machines provided as well as the network they run in, we can deploy a series of controls in order to keep a close eye and raise alerts if something odd ever happens.

These controls will allow us to go a long way, but there are still risks we cannot, and will probably never be able to control, like workers taking out their phones and taking a picture of their screen.

User Input

Since the machines are controlled by a VNC connection where we have control of the client and server, we can without much hassle keep a close eye on all the user input and actions. We can log and analyze both the keyboard key presses and mouse movements, or lack of them. With said data we can define what a fair use of those virtual machines look likes, which we can use to alert on and even block odd usage.

Network Communications

Having these machines on our network, we can also keep close eyes on the network traffic coming in or going out of them. This will allow us to insert things like transparent MITM proxies, IDS, IPS and whatnot and see all traffic going in or out of the machine. We could also keep control of DNS records and sites visited to act upon as with any of our networks, reducing the ability of a malicious worker to move freely around.

Session Recording

Thanks to machine emulators we can record the workers sessions in both screenshots or full video recordings of the screen. Since this happens on a virtual machine host level, it’s highly unlikely that a malicious actor can tamper with it, so it’s highly reliable forensics tool both for us and our clients if something ever goes wrong.

System and Application Logs

One extra piece of information that we can get from the virtual machine is system or application logs. That way we can, for example, see everything that has been output to the console of Chrome inside the VM, which can be used for DFIR or prevention, depending on the case.

Virtual Machine Lifecycle

A very important thing to consider is the lifecycle of the virtual machines used for performing the work. Current virtualization and cloud technologies allow us to create new machines without much effort, which we should use to our advantage. We should always keep a pool of newly created machines available for assignment to workers. Then, once a machine is assigned and

worker has been given access to it, we should start a reasonable timer for them to perform the work assigned, which shouldn't be more than 1 hour in most cases. Once that timer has reached the defined engagement time, or the worker has sent us the result of their work, we should store the remaining bit of information required (like logs or system information depending on what we are recording) and destroy the virtual machine. This will help us reduce infection window and keep access persistence to a minimum in case all of our controls fail.

Crowd Management

As with any type of engagement we will have start, middle and end to it, and relationship with workers is no different. They will come one day, start picking our work for some time and stop at some point, either on their own, or because they showed indications of poor professionalism or bad behavior and we need to stop allowing them to engage with us. We should make sure that workers engaged with our jobs are highly productive and trustworthy ones, and for that we can rely on different strategies.

Worker Scoring

Since we have a reliable, ongoing assessment of workers, we're able to look at a variety of factors to determine which workers we can rely on to provide quality results. Given these factors, we can define a worker as highly trusted, fairly trusted, training required, or not trusted.

One thing to notice here is that we should define a timeframe of assessments to use for the testers, taking records that are too old into consideration will favor new workers and penalize older and more experienced ones, when we want the opposite. The longer a successful engagement, the more we can trust a worker is a real, honest human being and not a bot or malicious actor.

Crowd-Manage the Crowd

The bigger our crowd of workers grows, the more time we will need to spend managing it and reviewing randomly chosen jobs to ensure nothing bad has happened with the data on them. This will, at some point, become unmanageable. For that we can rely on our most highly trusted workers to be a special force between the full crowd and us, reviewing more jobs that you can possibly review and flagging the ones that need your attention. Since this is a highly trusted task, a small, handpicked bunch of workers is recommended. You need to know each of the reviewers well enough to trust them with such an important task.

Special Requirements

Some customers will require each worker that engages with their data to sign an NDA or a contract of some sort, or have strict requirements about who can access their data. Since we

control the app the workers are going to use to perform the tasks, we can keep an extra set of requirements on our side to assign work. We can have them sign contracts for our customers, require them to provide a malware scan on their machine, or send pictures as proof of private working conditions. With those specific to us and with our customer requirements, we can define which tasks or data we trust a worker with.

Forum Monitoring and Worker Support

Most of the crowd providers have forums where the workers hang out. We also need to keep a close eye on those forums, as it's not uncommon for workers to post screenshots of tasks they have performed with questions about how to perform them, which of course, means our data could be included (and leaked) on them. To avoid this, it's advisable to have a strong support system for the testers where they can get the answers they need and don't find themselves forced to rely on each other. Correctly and continuously training them also helps reduce the chances of this happening.