Aircloak Challenge

Version 2, Oct. 16 2017

The Aircloak Challenge is a bounty program to better understand the properties of the Aircloak Insights anonymization system. Aircloak Insights is a system that sits between a database and an analyst. The database contains non-anonymized data about users. Aircloak Insights accepts SQL queries from the analyst and returns anonymized aggregate answers to the analyst. Aircloak Insights adds noise to answers, but does so in a way that is tailored to the semantics of the SQL. As such, Aircloak Insights operates both on the SQL and on the results returned by the back-end database.

Unlike traditional bounty programs, where attackers do penetration testing or examine source code for bugs, the Aircloak Challenge is looking for weaknesses in the design and implementation of its anonymization mechanisms. This is, to our knowledge, the first bounty program geared towards anonymization. The understanding gained from this program can lead to both stronger anonymization and to better risk assessment with respect to any weaknesses that exist in the system.

The attacker is given access to the SQL interface, and can make an unlimited number of queries to the system in an attempt to obtain information about single individual users in the database. Bounties ranging from $100 to $5000 (USD) are offered depending on how much individual user data can be learned, and on how much prior knowledge about the database is required.

We expect this challenge to be the first of multiple challenges. This challenge starts in November 2017 and runs for 6 months or until the total allocated budget of $15,000 is exhausted.

Bounties are paid through Paypal.

What a Program Participant Can Expect

The Aircloak challenge is a bounty program for a new and specialized technology. There may be a substantial effort involved in understanding the technology, designing attacks, and coding and executing the attacks. While there is always the possibility that a participant may quickly conceive a new killer attack, there is also every likelihood that a participant may devote several hours of background work and not come up with any new ideas. Ideally the participant is in any event interested in learning about the technology. In what follows, we outline the process that a participant can expect to go through after signing up for the bounty program:

- Read this document and understand the methodology for defining successful attacks (8 pages)
- Read the document describing the anonymization technology and the kinds of attacks it defends against (20-30 pages)
- Look over the discussion forum to learn from other attackers

- Sketch out an attack
- Confirm that the attack is not on the list of known resolved attacks or already discussed on the forum
- Communicate your attack sketch with us to learn if someone else is already pursuing it, or to learn for instance why we think the attack may succeed or fail (challenge@aircloak.com)
- Write a concise description of the attack and tell us about it so that you establish *bounty rights*
- Code and execute the attack, and measure its α,κ score (*effectiveness* α and *confidence improvement* κ) to determine the bounty amount
- Convey the attack code and attack results to us for validation

To Get Started

Sign up at www.aircloak.com/challenge. We will give you the necessary accounts to get started.

Basic Rules

By way of terminology, a *participant* is the person running attacks. The participant plays the role of an *attacker,* who tries to obtain individual information from the database through a series of malicious queries.

The participant is provided with:

- Login access to an Aircloak Insights system (API and web)
- A basic description of Aircloak's anonymization mechanisms (called Diffix)
- The full contents of the backend datasets (Aircloak provides 5 such datasets by default. The participant may provide their own datasets for attack.)
- A list of known resolved attacks
- A discussion forum open to all attackers
- One or more samples of attack source code

The participant may make an unlimited number of queries to Aircloak Insights. (Limited of course by the computing capacity of Aircloak Insights.) Using the answers, the attacker makes "claims" about the data of individuals in the dataset (see below for details). To the extent the claims are correct, the attacker may be paid a bounty. To receive a bounty, the participant must implement the attack and supply the attack code to Aircloak. Aircloak will independently validate the attack.

Duplicate Attacks

Aircloak does not pay for duplicate attacks. To receive a bounty, the participant must satisfy the following two criteria:

1. The attacker must establish *bounty rights* for a given attack
2. The attack must supply attack code that is validated by Aircloak

First *bounty precedence* goes to the first participant to adequately describe the attack to Aircloak. The description must be detailed enough that a third party could implement that attack given the description. Second bounty precedence goes to the second participant to describe the same attack, and so on. *Bounty rights* initially goes to the participant with first bounty precedence.

Upon receiving bounty rights, the participant must submit the attack code within two calendar months. If the participant fails to do so, then bounty rights are assigned to the participant with the next bounty precedence, and so on.

Aircloak reserves the right to determine what constitutes a duplicate attack. A given attack A may imply a class of closely related attacks A', A'' etc. For instance, suppose there is an attack that uses a certain condition clause, and that the condition may appear in a WHERE clause with or without a sub-query, or in a HAVING clause. These variants would all be considered a single attack, not multiple attacks, and accordingly would get a single bounty. In addition, if an attack A' is defended against by the solution to attack A, then attack A' may likely be considered to be in the same class as attack A, and therefore a duplicate. Having said that, Aircloak will work in good faith to resolve any dispute with the participant regarding duplication of attack.

The participant agrees not to publicly disclose the attack until Aircloak has had the opportunity to implement and deploy a defense. The participant allows Aircloak a maximum of 12 months to implement a defense.

Before spending significant time on an attack idea, the participant should of course look at the list of known resolved attacks. If the attack does not appear there, the participant is strongly encouraged to discuss the attack with Aircloak to establish bounty precedence, determine if another participant is already working on the attack, or to determine if there is an obvious flaw in the participant's idea.

A total of $15000 is allocated by Aircloak for the challenge. Bounties will be paid in order of submitted working implementations. If not enough money remains in the budget to pay a full bounty, then only the remaining budget will be paid.

Response Times

Aircloak will make every effort to satisfy the following response times:

- Time from submission of attack description to response (i.e. assignment of bounty precedence, indication that the attack is known resolved or unlikely to work, etc): **3 business days**
- Time from attack code submission to attack validation: **20 business days**
- Time from attack validation to bounty payment: **20 business days**

Attacks and Bounties (the α,κ score)

The goal of any attack is to *single out* a user. This is a criteria for anonymity according to the EU Article 29 Data Protection Working Party Opinion 05/2014 on Anonymisation Techniques (https://cnpd.public.lu/fr/publications/groupe-art29/wp216_en.pdf).

Singling out occurs whenever an attacker can determine, with high confidence, that there is a single user with one or more given attributes. For instance, the attacker may claim that there is a single user with attributes [gender = 'male', age = 48, zipcode = 48828, lastname = 'Ng']. If this is true, then the attacker has successfully singled out that user. The attributes don't need to be personal attributes as in this example. If the attacker correctly claims that there is a single person with the attributes [long = 44.4401, lat = 7.7491, time = '2016-11-28 17:14:22'], then that person is singled out.

Before executing an attack, the attacker may assume some *prior knowledge* of the database. In other words, the attack itself may exploit assumed prior knowledge on the part of the attacker (Note that in any event the participant is given a full copy of the dataset being attacked. This is for the purpose of determining whether an attack worked or not, and is not the assumed prior knowledge per se). For instance, in the previous example, the attacker might have prior knowledge that there are one or more users with attributes [gender = 'male', age = 48, lastname = 'Ng']. Assuming that the attacker does not have prior knowledge of the zip code, the goal of the attacker then is to make a claim about the value of the zip code where there is a single user with for instance attributes [gender = 'male', age = 48, zipcode = 48828, lastname = 'Ng']. In other words, the attacker needs to demonstrate that they can learn something new about a singled out user from the attack.

Confidence is a measure of the likelihood that an attacker's claim is correct. If 95% of an attacker's claims are correct, then the confidence of the attack is 95%. *Confidence improvement* (or just *improvement* for short) is the improvement in the attacker's confidence over a statistical guess. By way of example, suppose that 90% of the users in the database have zipcode = 48828 (which the attacker can learn directly from a query of the cloak). If the attacker knows through prior knowledge that there is a single user with attributes [gender = 'male', age = 48, lastname = 'Ng'], then the attacker can simply guess that this user's zipcode = 48828, and the attacker would be correct 90% of the time. If in an attack, the attacker gives a correct zipcode 90% of the time, then the attacker has not improved over a statistical guess. However, if the attacker correctly claims that zipcode = 48828 95% of the time, then the attacker has improved over a statistical guess. We measure improvement κ as `κ = 100*(C-S)/(100-S)`, where `C` is the attacker's confidence, and `S` is the statistical probability. In the previous example, the attacker's improvement is `50%` (`100*(95-90)/(100-90) = 100*5/10 = 50`).

In an attack, the attacker may have a certain amount of prior knowledge P. We measure P as the number of cell values known to the attacker. If the attacker knows all values for a given database column, and there are 5 million rows, then the attacker knows 5M cell values, and P=5M. If the attacker knows 5 attributes each for 500 users, then the attacker knows 2500 cells in total, and P=2500.

An attacker may learn new cell values (not known prior) when they successfully single out a user. We measure the amount learned L in an attack as the number of learned cells. For example, suppose that the attacker singles out 500 users, and learns two new cell values in each singling out. Then L=1000.

Every attack has an *effectiveness* score α. Clearly an attack is relatively more effective if less prior knowledge is required to carry out the attack. An attack is also relatively more effective if more cells are learned. We measure effectiveness α as `α=L/P`.

Bounties are based on a combination of effectiveness α and confidence improvement κ (the α,κ score). The highest bounty is $5000, and the lowest is $100. Bounties are assigned according to the following table:

From effectiveness value α	To effectiveness value α	Base Bounty
anything > 0	10^-11	$500
10^-11	10^-10	$600
10^-10	10^-9	$700
10^-9	10^-8	$800
10^-8	10^-7	$900
10^-7	10^-6	$1000
10^-6	10^-5	$1500
10^-5	10^-4	$2000
...	...	...
0.1	1	$4000
1	10	$4500
10	anything > 10	$5000

The base bounty is the bounty awarded if the confidence improvement is 95% or better. If the improvement is less than 95%, then the base bounty is multiplied by a factor `F` selected according to the following table:

Confidence Improvement κ	Factor F
95% - 100%	1
90% - 95%	0.9
80% - 90%	0.6
70% - 80%	0.3
50% - 70%	0.1
0% - 50%	0

If the factor `F` results in a bounty lower than $100, then the bounty is set at $100. In other words, $100 is the minimum bounty for any attack that achieves a non-zero bounty.

Datasets

Aircloak provides five datasets for the Aircloak Challenge. The datasets differ substantially in content, and so represent a wide variety of dataset types. They are described at the end of this document.

The participant is free to provide their own dataset, *but the dataset must be a real dataset for the bounty table to apply*. The dataset must have only a single UID-type column (this is the column that identifies the individual user in the dataset). Note also that currently we have not implemented protection for dynamic databases. The attack must take place on a static database. There may be other known restrictions on the dataset not mentioned here.

Of course, it must be legal for the dataset to be used, for instance because it is public, or because the data owners have approved the usage. If the participant wishes to use a synthetic dataset or a small dataset, then please contact us to discuss (challenge@aircloak.com).

Prior Knowledge

The attacker is allowed prior knowledge because it may often be the case in real life that the attacker knows certain details about the dataset, or about users in the dataset through external sources. For instance, in 2002, Sweeney was able to attack an "anonymized" medical database using knowledge obtained in a publicly available voter registration dataset. It may also be that an analyst has full access to a dataset that was joined with another dataset, giving them substantial knowledge about portions of the resulting dataset.

It is up to the participant to determine what prior knowledge they would like to assume in the role of an attacker for any given attack. For the bounty table to apply, however, the participant cannot use direct knowledge of cell values in the dataset in determining the attacker's prior knowledge. The attacker may, however, use knowledge learned from queries to the database made via Aircloak.

Following are a few examples of how a participant may select prior knowledge without exploiting direct knowledge of the contents of the dataset:

- Select one or more columns (a *column group*). Select some number of random rows, and define the cells in those rows for the column group to be prior knowledge. The number of random rows can be a fraction of all rows (e.g. 50%), or an absolute number (5000 rows, or all but 5000 rows).
- Select a column group. Select some number of random users for which all column values for those users are prior knowledge.
- Select a column group. Select all rows (or all users) where some column has a certain value.
- Some combination of the above. For instance, select all columns for 50% of the users. Additionally select from the remaining 5000 rows where lastname, age, and marital_status are known.

If the participant is unsure as to whether their method of defining prior knowledge is acceptable, please contact us at challenge@aircloak.com. If the participant has an attack in mind that does require very specific prior knowledge based on knowledge of the dataset, please contact us to discuss. We would be interested in understanding such attacks even if they won't exist in practice, and may agree on some bounty.

Establishing Confidence and Improvement

Confidence is a measured statistical property: the attacker generates a series of singling out claims, and each claim is either correct or incorrect. The confidence `C` for the attack is the number of correct claims divided by the total number of claims (`C=correct/total`).

As stated already, confidence improvement is defined as `100*(C-S)/(100-S)`, where `C` is the attacker's confidence, and `S` is the statistical probability of a value or set of values. `S` is computed as the average statistical probability across all claims. In other words, `S=sum(S_i)/N`, where `S_i` is the statistical probability of claim `i`, and `N` is the number of claims.

`S_i` for any given claim is computed as `S_i=C_i/R`, where `C_i` is the count of the number of rows that match all of the unknown claim conditions, and `R` is the total number of rows.

To give an example, suppose that the attacker knows [gender = 'male', age = 48, lastname = 'Ng'], and makes a claim [gender = 'male', age = 48, lastname = 'Ng', zipcode = 12345, firstname = 'Bao']. `C_i` then is the number of rows where [zipcode = 12345, firstname = 'Bao'].

Note that the statistical probability here is derived from all claims, not only those where the attacker happens to be correct.

To get a reasonably accurate confidence value, we require at least 200 claims. If the participant cannot for some reason produce 200 claims, but can argue that the attack confidence is never-the-less high, then please contact us to discuss what a fair bounty should be.

If a single attack (i.e. an instance of an attack with a given dataset and assumed prior knowledge) cannot produce 200 claims, then the attacker may need to replicate the attack with different starting conditions (different prior knowledge or dataset). For instance, if the attack requires prior knowledge of all cells but one, then the attack cannot make more than one claim (on the single unknown cell). Such an attack would have to be replicated 200 times with a different unknown cell each time.

It may be possible in some cases that the attacker knows that certain claims are more likely to be correct than certain other claims. In these cases, the attacker may choose not to make claims on those that would otherwise be low confidence. For instance, suppose that the attacker uses three complete columns as prior knowledge on a dataset with 1M rows (so P=3M). Suppose that of the millions of possible claims the attacker could make, the attacker knows that 500 of them are likely to be high-confidence claims. The attacker then only needs to make those 500 claims. Supposing that all of them are correct, then L=500, confidence C=100%, and improvement κ=100%. Note however that in computing effectiveness α, we still use the full prior knowledge of P=3M.

Datasets

Aircloak provides the following datasets. All datasets are publicly available and are used in the original form (not cleaned) with the exception that a `lastname` column has been added to each dataset. The `lastname` is synthetic generated from the distribution of last names published by the US census bureau.

This column has been added to give each dataset another kind of text data field that might be interesting to attack.

In the following descriptions, we refer to online documents that more-or-less describe the meaning of the data in the columns. Some of these descriptions are less than perfect, but understanding of the column meaning is not important for an attack. Other than the column that contains the unique User ID, Aircloak Insights does not differentiate between columns, for instance whether a column is sensitive or not, or whether a column contains personally identifying information. Likewise neither does the α,κ score distinguish between columns. The goal of the attacker is simply to single out users, regardless of the meaning of the data used to single out.

The scihub dataset:
This lists downloads from the scientific paper pirating website sci-hub.io for the month of September 2015. It has 8 columns and roughly 600K distinct users (downloader IP address, hashed) and 5M rows (downloads). Columns include user ID, document ID, timestamp, and location (city and country, and corresponding latitude and longitude).

The taxi dataset:
This is one day of taxi rides for all of NYC (January 8, 2013). It has 22 columns and roughly 25K distinct users (drivers) and 500K rows (rides). The med (medallion) identifies the car, and the hack identifies the driver. The dataset is configured to protect the hack. Links to documents describing most of the other column definitions can be found at http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml.

The banking database:
User data and banking transactions for a real bank in the Czech Republic (this data is already slightly pre-anonymized, but retains the structure of the original data). It has 7 tables with between 4 and 18 columns for roughly 5000 distinct users (customers). There are roughly 1.25M transactions, 800 loans, and 8000 orders, among other things. A description of the data is linked from http://lisp.vse.cz/pkdd99/.

The census datasets:
Actual US census data (pre-sampled by the Census Bureau, but otherwise not anonymized). There are 112 columns with roughly 3M individuals from 1.4M households. We have configured two datasets from this data. One dataset protects the individuals. The other protects the household. Column definitions can be found via this page: https://usa.ipums.org/usa-action/variables/group.