Technical Report MPI-SWS-2021-002

Paul Francis

Feb. 2, 2021

Procedures and Rules for the 2020 Diffix Bounty Program

The purpose of this technical report is to document the rules of the Diffix anonymity bounty program that was run in 2020. This is the second time the bounty program was run, and contains modifications to how the bounty was computed.

This tech report duplicates the rules of the bounty program rules exactly as they were published at the URL https://www.gda-score.org/mpi-sws-diffix-challenge-2020/ in 2020.

The specification of Diffix Dogwood and associated attacks are in a separate tech report (MPI-SWS-2021-001).

MPI-SWS/Diffix Bounty 2020

May 29, 2020 | Leave a comment

Sponsor	MPI-SWS
Total purse	$30,000
Maximum award per attack	$10,000
Start date	May 26, 2020
Duration	6 months
Anonymization	Diffix Dogwood
Implementation	Aircloak Insights
Contact email	diffix-bounty@gda-score.org
Document Version	V3, May 29, 2020

Documentation and Resources

The following documents describe the anonymization mechanism (Diffix Dogwood), and a known set of attacks that Diffix Dogwood is able to defend against:

- **Specification of Diffix Dogwood and attacks**

The Github repository at github.com/gda-score/code contains a python package for running attacks and computing a score. Credentials may be requested at diffix-bounty@gda-score.org. The following two GitHub files contain well-documented examples of attacks for Singling Out and Linkability. Participants can use these as a template for their own attacks:

- Singling Out attack
- Linkability attack

This document provides more background on how to use the GDA Score code library to generate the GDA Score for a given attack:

- **Guide to writing attacks**

The following is an online system where the participant can experiment with the Aircloak implementation of Diffix Dogwood:

- **Diffix online onboarding system**

The databases provided by the bounty program can be explored here:

- **SQL online interface to databases**

This is the customer documentation for the Aircloak implementation of Diffix Dogwood:

- **Online documentation of Aircloak Insights**

As background material, the following two documents describe the GDA Score. The scoring for this bounty programmed, described below, is based on the GDA Score.

- Overview of the GDA Score
- Description of the GDA Score

Finally, you may read comments on this page from other users, or make comments yourself.

What a Program Participant Can Expect

This is not a typical pen test bounty program. The goal is to find vulnerabilities in the Diffix Dogwood data anonymization technology, and for us to openly publish these vulnerabilities to further scientific research in data anonymization.

Diffix Dogwood is a specialized technology, and there will likely 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 doing this out of academic interest as much as a means of making money.

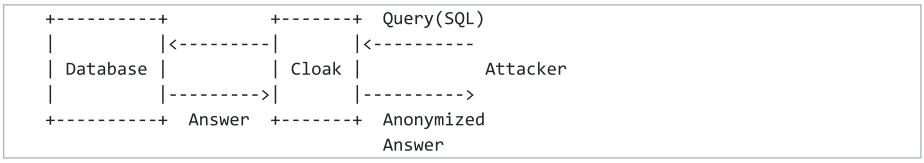
The process that a participant can expect to go through for the bounty program is:

- Read this document and the above documentation.
- Sketch out an attack
- Discuss your attack sketch with us at diffix-bounty@gda-score.org to learn if someone else is already pursuing it, or to learn for instance why we think the attack may succeed or fail
- Write a concise description of the attack and tell us about it so that you establish *bounty rights*
- Code and execute the attack using the GDA Score library, and produce a GDA Score
- Convey the attack code and attack results to us for validation

Feel free to leave comments on this page if you believe they may be of general interest.

Basic Rules

By way of terminology, a *participant* is the person participating in the bounty program. The participant plays the role of an *attacker,* who tries to obtain individual information from the database through a series of malicious queries to the cloak (Aircloak Insights), which is running Diffix Dogwood. The *sponsor* is the Max Planck Institute for Software Systems (MPI-SWS) (also refered to as “we” or “us” in this document).



The participant is provided with:

- Login access to a cloak system (API and web).
  - (Generic access is provided automatically with the GDA Score attack library.)
- A complete description (best effort) of Diffix Dogwood.
- A description of known attacks.
- The full contents of the datasets being protected by Diffix Dogwood (see db001.gda-score.org).
- Example attacks demonstrating how to use the GDA Score attack library (python).

The participant, acting as an attacker, may make any number of queries to the cloak (limited of course by the computing capacity of the cloak). Using the answers, the attacker makes *claims* about the data of individuals in the dataset (see Description of the GDA Score 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 the sponsor. The sponsor will independently validate the attack.

The code must be based on the GDA Score library unless there is a compelling reason not to. If the participant does not wish to use the GDA Score library, he or she must get approval from us in advance.

This is a white-box bounty program. The sponsor has made a good-faith attempt to describe Diffix Dogwood adequately for attackers to understand the system and formulate attacks.

Duplicate Attacks

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

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

First *bounty precedence* goes to the first participant to adequately describe the attack to the sponsor. 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. The timestamp associated with each bounty precedence set as the time that an *adequate* description is received by the sponsor via the mailbox diffix-bounty@gda-score.org.

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.

The sponsor 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, the sponsor 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 is strongly encouraged to discuss the attack with the sponsor 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 budget of $30,000 is allocated by the sponsor 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

The sponsor will make every effort to satisfy the following response times:

- Time from submission of attack description to response (i.e.Â 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**

Confidence Improvement and Prior Knowledge

The dataset being protected by Diffix Dogwood is referred to as the *protected dataset*. As mentioned above, the participant is given a complete copy of the protected dataset. The purpose of the copy is so that the participant may determine the effectiveness of any given attack.

The *attacker* (i.e.Â the participant in the role of attacker) of course does not have access to the dataset copy. The attacker may, however, have partial knowledge of the protected dataset. We refer to this as *prior knowledge*, and use it to emulate knowledge that the attacker may have obtained through external means. Many real attacks exploit prior knowledge.

The GDPR criteria for anonymity is defined according to the EU Article 29 Data Protection Working Party Opinion 05/2014 on Anonymisation Techniques. This opinion defines three criteria, or risk factors, *singling-out*, *linkability*, and *inference*. The bounty program tests for all three of these risk factors.

For the purpose of this bounty, singling out occurs whenever an attacker can determine, with high confidence, that there is a single user with one or more given attributes. With singling out, a claim is of the form: **"There is a single user with the following 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.

Inference assumes that the attacker has prior knowledge of one or more attributes for one or more rows, and that additional (unknown) attributes are inferred from the known attributes. With inference, a claim takes the form: **"All rows with attributes A, B, and C also have attribute D"**, where A, B, and C are known, and D is unknown. For instance 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 for this user or users, for instance attributes [gender = 'male', age = 48, lastname = 'Ng', zipcode = 48828]. In other words, the attacker needs to demonstrate that they can infer something new about the user or users from the attack. The claim is correct if all users have the claimed attributes, and incorrect if any one user does not.

Note that inference for a single user is essentially the same as singling out.

To test linkability we provide a second dataset (*the linkability dataset*) which is linked with the protected dataset. The linkability and protected datasets share some set of users, but each also contains users not in the other. For the purpose of the challenge, the linkability and protected datasets both contain all

rows for the common users. A linkability claim is of the form **"This user in the linkability dataset is also in the protected dataset."**

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 for an inference attack, 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 CI as `CI = 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`).

> Note that the Article 29 definition of inference does not include the notion of confidence improvement, rather only of confidence. However, the purpose of the inference risk is in large part to capture an attack on K-anonymity, whereby simply knowing which k-anonymity group a user belongs to can reveal additional attributes with 100% certainty. Given that the whole purpose of a database system, even an anonymized one, is to give analysts statistical knowledge, we believe that confidence improvement is the appropriate metric for inference.

We measure the amount of prior knowledge PK 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 PK=5M. If the attacker knows 5 attributes each for 500 users, then the attacker knows 2500 cells in total, and PK=2500. The GDA Score library provides a mechanism, the `getPriorKnowledge()` API call, for providing prior knowledge on request and computing PK.

Bounty computation

The maximum award for any given attack is $10,000. This is divided into two parts of $5000 each:

- **Effectiveness (E):** How well an attack works against **any** data. Effectiveness is measured as how much can be learned relative to how much prior knowledge the attacker has.
- **Coverage (C):** How well an attack works against **all** data.

In essence, effectiveness measures whether any data at all is at risk, and coverage measures how much data is at risk.

The participant must complete both the effectiveness and coverage parts to receive the prize for either part.

Effectiveness computation

For singling out and inference attacks, any attribute (cell) values claimed in the attack that are not known prior are *learned cell values*. For these attacks, we measure the amount learned L 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.

For linkability attacks, the attacker learns that a user in the linkability dataset exists in the protected dataset. We measure the amount learned L as the number of users claimed to exist in the protected dataset. If an attacker claims to link 500 users, then L=500.

From this we compute two *learned to prior knowledge ratios* `LPK_one` and `LPK_all`. The more that can be learned for a given amount of prior knowledge PK, the more effective the attack. `LPK_one` is the LPK required to learn a single cell for a single user, whereas `LPK_all` is the LPK required to learn all that can be learned about multiple users. The effectiveness E is computed as the average of the two LPK measures: `E = (LPK_one + LPK_all)/2`.

The reason for two separate LPK measures is as follows. It may often be the case than an attacker wishes to attack a specific person that the attacker knows (victim), and for which the attacker has prior knowledge. For instance, an attacker may know that a colleague has traveled to certain places at certain times, and wants to use this as prior knowledge in an attack. However, certain types of attacks only work if the attacker has prior knowledge about a group of individuals. Both of the successful attacks in the first bounty, for instance, required that a group of users to run the attack.

In the case where the there is a single victim known to the attacker, an attack whereby the attacker requires prior knowledge of the single victim is much more likely to succeed than an attack that requires prior knowledge of a group of users. An attack that requires prior knowledge of only a single victim receives a higher bounty than an attack that requires prior knowledge of a group of users.

`LPK_one` is the measure of how much learned to prior knowledge ratio to attack a single cell for a single user. `LPK_all` is the measure for attacking as many cells for as many users as possible given the prior knowledge.

The reason we limit `LPK_one` to a single cell is to reflect the fact that typically an attacker is interested in learning specific values: other values, even if they can be learned, are of less interest. On the other hand, it may also well be the case that the more learned, the better (for the attacker). We capture this by measuring everything learned for `LPK_all`. Given this, `LPK_all` is computed as `LPK_all = L/PK+1`, and `LPK_one` is computed as `LPK_one = 1/PK+1`.

For example, suppose that to run an attack, the attacker needs to have prior knowledge of two cell values for 100 users. In this case, PK=200. Suppose that from this attack, the attacker can learn 4 new cell values for each of the 100 users. Therefore L=400. LPK is then computed as follows:

```
LPK_one = 1/(PK+1) = 1/201 = 0.005
LPK_all = L/(PK+1) = 400/201 = 1.99
E = (LPK_one + LPK_all)/2 = 1.00
```

From effectiveness E, we compute a base bounty BB according to the following table:

From effectiveness value E.	To effectiveness value E.	Base bounty BB.
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 BB is then adjusted by the confidence improvement CI: the higher CI is, the higher the bounty. The adjustment is a factor F taken from the following table:

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

The effectiveness bounty EB is computed as `EB = BB * F`

All of the values required to measure EB are computed automatically by the GDA Score library.

Coverage computation

Effectiveness is measured against whatever victims the attacker chooses (subject to what is regarded as legitimate prior knowledge and the need to obtain enough attack samples to establish an accurate measure.) Obviously it is in the participants best interest to attack the most vulnerable data.

Coverage is measured as the average effectiveness across all data. To compute the coverage bounty, an effectiveness bounty EB is computed for each column, where the attack is against an arbitrary set of cells rather than against the most vulnerable cells. The coverage bounty CB is then computed as the average EB taken across all columns for the given table being attacked.

We do not precisely define here what we mean by an arbitrary set of cells. Ideally the cells are somehow selected randomly, but if this is not possible the participants and the sponsor need to agree on what this is. The intent in any event is to determine roughly are much an average cell of data is at risk.

Minimum total bounty

If the total bounty resulting from both parts is above $0 but less than $100, the bounty is set at $100.

GDA Score attack output

The GDA Score attack library generates a `.json` file that contains the results of the attack. The confidence improvement score CI can be found at `'score':'scores':[0]:'confidenceImprovement'`.

Datasets

This bounty uses the four datasets provided by the GDA Score project. The datasets differ substantially in content, and so represent a wide variety of dataset types. They are described here.

For singling out and inference attacks, the full dataset is used as the protected dataset. These are the databases `raw_banking`, `raw_census`, `raw_taxi`, and `raw_scihub` at <db001.gda-score.org>.

For linkability attacks, a protected linkability dataset and prior-known linkability dataset are provided. The protected linkability datasets are `raw_banking_link`, `raw_census_link`, `raw_taxi_link`, and `raw_scihub_link`. The prior-known linkability datasets are `raw_banking_link_pub`, `raw_census_link_pub`, `raw_taxi_link_pub`, and `raw_scihub_link_pub`.

The participant is free to provide their own dataset, *but the dataset must be a realistic dataset for the bounty table to apply*. In other words, the dataset cannot be custom designed to be attackable, but otherwise not represent a realistic dataset. 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.

Selecting 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 in part using knowledge obtained in a publicly available voter registration dataset.

We emulate prior knowledge by allowing the attacker to have knowledge of data in the protected dataset. The GDA Score library provides a mechanism whereby the attacker can request prior knowledge. The mechanism computes the prior knowledge score PK. This mechanism is designed to emulate the kind of prior knowledge an attacker might have in real life.

By way of explanation, suppose that a given attack requires certain data from the database, for instance a group of at least 30 users where 29 of the users have a certain column value, and the 30th user has a different column value. Suppose, however, that this condition is relatively rare, occurring for only 1 in 10000 users. An attacker would therefore on average need to have prior knowledge of 10000 users in order to be able to find the 1 attackable user.

To properly reflect this reality, the attacker would need to request prior knowledge for roughly 10000 users, examine this knowledge for an exploitable user, and then run the attack. Even if only the one user is attacked, the fact that prior knowledge about 10000 users was required to find the one user needs to be accurately reflected in the PK value.

The GDA Score mechanism `getPriorKnowledge()` allows the attacker to specify a set of columns whose values he or she would like to learn, and then to request rows for those values according to the following criteria: * A fraction of all rows selected randomly * All rows for a fraction of all users selected randomly * A specific number of rows selected randomly * All rows for a specific number of users selected randomly * All rows whose value for a given column fall into a specified range * All rows whose value for a given column match a specified set of values

The `getPriorKnowledge()` mechanism can be called multiple times, but if so the resulting PK measure may not be accurate, because the same data can be measured multiple times. In this case PK must be measured by hand. (Note that the `getPriorKnowledge()` API must be used rather than the `askKnowledge()` API.)

Public Knowledge

It is of course reasonable to assume that there exists public knowledge about a database. Certainly the column names and types are publicly known. In addition, common values for certain columns may be known. For instance, one could assume that the values for the ‘gender’ column are known to be ‘M’ and ‘F’. Public knowledge is not counted in the PK score.

For the purpose of this bounty program, we assume that anything learned from Diffix Dogwood itself may be regarded as public knowledge.

Establishing Confidence Improvement

To get a reasonably accurate confidence value for the effectiveness score, 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.Â 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 in order to improve the effectiveness or coverage scores. For instance, suppose that the attacker uses three complete columns as prior knowledge on a dataset with 1M rows (so PK=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 CI=100%. Note however that in computing effectiveness E, we still use the full prior knowledge of PK=3M. In computing coverage C, only claimed cells are counted as learned cells.

Leave a comment

Your email address will not be published. Required fields are marked *

Your name *

Your Email address *

Your comment

Post Comment

← Diffix Dogwood Specification and Attacks