Anonymous Local Count Statistics Using PCSA

Samir Menon, GSOC 2017, Tor

Currently, pyobfsproxy keeps a list of IP's in memory to compute the `"from %d unique address(es)"` message in the heartbeat. A similar strategy is employed by `src/or/geoip.c` in computing the heartbeat statistics. See tickets #7532 and child #15469.

We would prefer not to do this, since it allows an attacker to retrieve all of the client IP addresses if they compromise a node. Our goal is to build a system that can continue to provide the statistics we currently do, but without revealing client IPs if a node is compromised.

Alternatives

I'll explain why some other privacy-preserving strategies are unsatisfactory.

One simple remedy would be a Bloom filter of hashed IPs. That would allow us to perform a simple lookup each time we see a client to check if they have connected before. If we haven't seen their IP before, we'll increment the unique users counter by one.

Unfortuantely, this method is critically vulnerable to a brute force attack. Since there are only 32 bits in the IPv4 space, an attacker can simply check all IP's and get the set of client IP's back. Even if we all used IPv6, an attacke could still confirm with a fairly high probability that a specific IP had connected to the node.

The remedy for this that the EFF's cryptolog uses is a salt that changes daily, but this allows an attacker to determine (or confirm) IP addresses that have connected on the same day. We would prefer a system that reveals none of the IP addesses.

Solution

PCSA

We can use a well-studied method of computing the number of unique elements seen without actually storing every seen element called "Probabilistic Counting with Stochastic Averaging", or PCSA.

The principle of the technique is quite simple, so I'll explain my understanding of it it here, but there are many resources online for learning more.

Firstly, let's define the 'find first set' bit operation: it reports the position of the least significant '1' in a bit string, or throws an error if the bitstring is 0. This is easy and fast to compute. We'll refer to it as f.

Then we can note that given a random bitstring b of some arbitrary length, the probability that $f(b) = 0$, that is that b ends in a 1, is 2^{-1}. Furthermore, the probability that $f(b) = 1$ is 2^{-2}, and in general,

$$P(f(b) = n) = 2^{-n-1}$$

We can use this in PCSA as follows:

Say we have one bitstring that we keep in memory, which we'll call a bitmap. Then each time we see an element b, we'll compute $f(b)$, and then set that position in the bitmap to 1. In other words, we'll find the position of the first 1 bit of the element, and then set to 1 the bit in that same position in the bitmap.

Assuming the elements are randomly distributed, the more unique elements we see, the larger the value of the bitmap will get. This happens because after seeing a lot of bitstrings, we will probably have seen at least with a large number of 0's before the first 1, and so the bitmap will get bigger. This simulation in shows this happening live, and these slides explain it in more detail.

There is more subtelty involved in the actual algorithm. We can perform 'stochastic averaging' with multiple bitmaps to get a more accurate estimate, and there are various correction constants that need to be applied to ensure the best estimate.

With those improvements, we have mathematical evidence that the estimate the algorithm provides will be close to the actual number of unique elements seen.

Refinements

There are two remaining questions:

Firstly, how can this work for IP addresses, since they are not randomly distributed? The resolution here is to just apply a simple hash function (like Jenkins hash) to the IP's, so that we get the accuracy guarantees proved in Flajolet's paper.

Secondly, does this reveal anything to the attacker? Very importantly, no, it does not. The beauty of PCSA (which I only fully understood thanks to Jaskaran on tor-dev) is that from the just the bitmap, an attacker can tell essentially nothing about which IP's have been seen. Even assuming that they can fully reverse the Jenkins hash given a set bit k an attacker can only know that one of 2^{32-k-1} IPv4 addresses have been seen. I'd like to find a more formal proof of the attacker's inability to recover IP's eventually, but this will suffice for now.

Implementation Timeline

There are 12 weeks of programming for GSoC 2017, from May 20th - August 20th.

Weeks 1-2

Implement the simplest version of PCSA in `obfsproxy/common/heartbeat.py`, along with unit tests and documentation of the feature. This version will use only one bitmap (no correction/hashing). This will mostly serve as a sort of 'dry run', getting used to conventions used in the Tor project.

Weeks 3-4

Implement the full-fleged version of PCSA in obfsproxy, using multiple bitmaps, Jenkins hashing and correction factors.

Week 5

Test both versions on a real-world set of IPs (will have to look into finding data), measuring error rates, and comparing them to theoretical error estimates. Produce a document that shows these results.

Week 6

Evaluate which of the dependencies on the clientmap structure can be removed, and which we should try to compute using PCSA. From #15469, it seems like we should use PCSA to replace the heartbeat log line, and remove actual IP addresses from the clientmap, instead just storing country and version information whenever we see a new client. Need to hammer out exact details in consultation with the community.

Week 7-10

Replace statistics provided in /src/or/geoip.c with computed ones using PCSA, as per ticket #15469. Since this is more complex codebase, and in C rather than Python, I expect this to take longer and involve more testing than previous work.

Week 11-12

Test and document the changes made to /src/or/geoip.c, and confirm that bridge statistics still work correctly.

Since some of these things may take more or less time, if I have leftover time, I'll spend it looking for other places in the Tor ecosystem where we could use PCSA to keep less sensitive data in memory.

Point us to a code sample: something good and clean to demonstrate that you know what you're doing, ideally from an existing project.

I wrote a TLS 1.2 client for Prof. Dan Boneh's cryptography class. The spec is here

Why do you want to work with The Tor Project in particular?

I support internet freedoms, specifically privacy, and I can program. One of the best ways for me to advance that cause is by working on Tor. It certainly seems more productive than my current strategy, which is to badger my friends into caring about it :)

More personally, I think Tor is really cool. I'd love to be able to tell my friends that I spent my summer working on (what I think is) one of the most important internet technologies of our time. I think I'd walk away with a lot of experience that would be difficult to get anywhere else.

Tell us about your experiences in free software development environments. We especially want to hear examples of how you have collaborated with others rather than just working on a project by yourself.

I have worked as a backend web developer at three different companies, where my code was always proprietary I found that stifling, and chose to instead spend last summer doing Computer Science research in the Stanford Logic Group. I liked that my work would (in the form of freely availible papers) be public. From my time in research and as a web developer, I have experience communicating with other software developers and users, using bug trackers, writing quality documentation, and working with source control. I want to use GSoC to start working in a free software environment, because I think it will make a happier and more productive programmer! :D

Will you be working full-time on the project for the summer, or will you have other commitments too (a second job, classes, etc)? If you won't be available full-time, please explain, and list timing if you know them for other major deadlines (e.g. exams). Having other activities isn't a deal-breaker, but we don't want to be surprised.

I will be working full time on the project. I have two exams between June 9-12. I have a small (<8 hours/week) teaching role for 8 weeks (June 26 - August 18), where I teach introductory Computer Science.

Will your project need more work and/or maintenance after the summer ends? What are the chances you will stick around and help out with that and other related projects?

I think my project is essentially self-contained; it's a one-time improvement to how we compute statistics based on IP. I also think it is feasible to get an actual patch done by the end of the summer, so I don't think it shoudl take more work. Certainly it would be good to keep this PCSA spproach for the future, and expand it to other kinds of statisitics that we currently compute, if any.

I'd love to keep working on Tor after the project ends. Next school year, I think I'll have a little more time to devote to programming. In particular, I'd love to poke around Tor Messenger, since I have a fairly large amount o Javascript and Python experience.

What is your ideal approach to keeping everybody informed of your progress, problems, and questions over the course of the project? Said another way, how much of a "manager" will you need your mentor to be?

I generally prefer to be the one who informs others, rather than waiting to be asked questions by a 'manager'. I'll inform the community on my progress by writing updates and sending them via tor-dev whenever I complete a major piece of the project, and more casually, by hanging out on IRC. I've loved chatrooms from my earliest days on the internet; I think that being availible on IRC will help me build a good relationship with fellow Tor developers.

What school are you attending? What year are you, and what's your major/degree/focus? If you're part of a research group, which one?

I'm at Stanford University, studying Computer Science. I've focused in applied cryptography and computer security; I've taken cryptography, internet security, and Bitcoin classes, all taught by Prof. Dan Boneh. I did research over the summer of 2016 in the Stanford Logic Group, with Prof. Michael Genesreth, building computational logic systems that non-programmers could use.

How can we contact you to ask you further questions? Google doesn't share your contact details with us automatically, so you should include that in your application. In addition, what's your IRC nickname? Interacting with us on IRC will help us get to know you, and help you get to know our community.

My IRC nickname is 'samir2'. The best way to contact me is email; I respond the fastest to menon.samir@gmail.com. We could also arrange a call using something like Google Hangouts.

Are you applying to other projects for GSoC and, if so, what would be your preference if you're accepted to both? Having a stated preference helps with the deduplication process and will not impact if we accept your application or not.

No, not as of submitting this.

Is there anything else that we should know that will make us like your project more?

I like talking to people about PGP, Signal, and Tor a lot! Like, not my fellow programmers, but general audiences I think it's really important that we let people know that these are technologies that *anyone* can use. I'm fairly politically active - my friends and I have cofounded the Stanford Political Union; we oragnize civil debates on controversial policy issues (like healthcare and automation).