From Hacking Botball Controllers to Having My Code in the Number One Privacy Web Browser
(Or: What a 2011 Botball Alumni Is Doing in 2022)
Jeremy Rand
The Namecoin Project / Norman Advanced Robotics Alumni / Team SNARC Alumni
jeremy@namecoin.org

From Hacking Botball Controllers to Having My Code in the Number One Privacy Web Browser (Or: What a 2011 Botball Alumni Is Doing in 2022)

1 Introduction

The Botball Educational Robotics Program may look to an uninformed outsider like just a competition, but in reality, it serves as an excellent preparation for the real world of programming (both robotics and non-robotics). I'm one of the Botball alumni who is now out in the real world doing software development, in the privacy/security/cryptography sector. This paper is a case study of the work I'm doing now with Tor Browser and Namecoin; it is hoped that this paper will contribute to the broader understanding of why Botball is so important as an educational program, both in terms of the success stories of Botball alumni, and the specific ways that Botball prepares its students for that success.

This paper can be considered a sequel to my GCER 2017 paper "Making HTTPS and Anonymity Networks Slightly More Secure" [1], but both that paper and this one can be read independently.

2 My Background in Botball

I was a Botball student between 2003 and 2011 at Whittier Middle School and Norman Advanced Robotics in Norman, OK. After graduating from Norman Advanced, I founded and led Team SNARC, which participated in KIPR Open and KIPR Aerial until 2015 when I finished my bachelor's degree in computer science. Norman Advanced and Team SNARC won quite a few awards while I was there, in large part due to my interest in hacking the Botball controllers (I published GCER papers on hacking the XBC, CBC, Link, AR.Drone, and Create); Norman Advanced continues to have a stellar reputation to this day. While leading Team SNARC, I also mentored three middle-school Botball teams at Alcott Middle School and Whittier Middle School.

I joined the Namecoin development team in 2013, becoming full-time in 2018 when I finished my master's degree.

3 Background: Privacy and Metadata

When most people think of online privacy, they're probably thinking about encryption, which hides the *content* of their communications. However, while encryption is important, this misses a more important aspect: *metadata*. The term "metadata" literally means "data about data", and it covers everything about your communication *other than* the actual text. For example, all of these are metadata:

- Who sent the message?
- Who received the message?
- When was the message sent?
- When did the recipient open the message?
- What was the size of the message?
- What software was used to send and receive the message?
- From what location was a message sent or received?
- Were two different messages sent by the same person, even if we don't know who they are?

Metadata turns out to be much more useful than content for attackers who want to violate your privacy. Why? Processing content requires natural language parsing, which is both error-prone and expensive (try using Google Translate to get an idea of how finicky natural language parsing can be). In contrast, metadata usually can be processed using simple algorithms from machine learning and graph theory; this is generally a lot less error-prone and is cheap enough for mass surveillance (whether by an intelligence agency or a surveillance capitalist).

As an example of what metadata can reveal about you, imagine the following sequence of events:

- Alice receives a phone call from a medical diagnostic clinic specializing in a rare, sometimes-fatal disease.
- Alice calls her cousin that night; the conversation lasts hours, which is unusual for Alice and her cousin.
- Alice makes a financial transaction to a medical treatment center, which also specializes in that disease, the next day.

It presumably goes without saying that this metadata can tell you a good deal about Alice, despite not knowing the text of what was actually said in those phone calls or what the financial transaction was billed for.

Or, if you don't want to take my word for it, here's former NSA General Counsel Stewart Baker [2]:

"Metadata absolutely tells you everything about somebody's life. If you have enough metadata, you don't really need content."

Or, for a more dramatic summary of what's at stake, here's former NSA Director Michael Hayden [2]:

"We kill people based on metadata."

Unfortunately, while metadata is much easier for attackers to process, it is also much more difficult to hide than content. Encrypting the content is easy, but how do you stop your ISP or your email provider from knowing who sent a message? To a large extent, the reason why content encryption gets more attention is simply because it's easier, not because it's the most important protection.

4 Tor Browser

The current leading project to protect metadata privacy is Tor Browser [3]. Tor Browser does two main things: it hides the sender and recipient IP address from each other (and from their ISP's), and it makes all instances of Tor Browser look the same, so that metadata called a *browser fingerprint* (e.g. which fonts you have installed) cannot be used to determine whether two different pieces of communication are associated with the same user.

Protecting the IP address is done via a technique called *onion routing* ("Tor" is an acronym for "The Onion Routing" or "Tor's Onion Routing"), which was originally invented by Paul Syverson of the U.S. Naval Research Laboratory. (Syverson and NRL are still involved with Tor research and development today.) Onion routing works by wrapping the communication in multiple layers of encryption (like the layers of an onion); the communication is then bounced through a sequence of relay servers, where each relay decrypts one layer and passes the communication to the next relay. The layered encryption prevents the first relay from knowing anything about the recipient, and prevents the final relay from knowing anything about the sender; all they know is which relay is adjacent in the chain. As long as the relays aren't all colluding, this allows two parties to communicate with each other without knowing each other's IP address, and prevents any intermediary such as an ISP or one of the relays from knowing who was talking to whom. Tor uses 3 relays for each party who wants to be anonymous; thus, if both the sender and receiver of a communication want to be anonymous, Tor bounces the communication through a sequence of 6 relays.

5 Usability Issues with Tor Browser

Tor Browser provides excellent privacy, but this comes at a mild cost in usability. The most obvious cost is in speed: the relays are all run by volunteers, and bouncing communications through 6 relays will necessarily incur a nontrivial slowdown in terms of both bandwidth and latency. For example, while residential Internet connections routinely reach 10 MiB/s (often much higher) when downloading a large file, downloading a file over Tor usually only can reach around 500 KiB/s. The latency cost is most noticeable when doing real-time activities, e.g. voice calls can feel unnatural due to the extra delay, and real-time competitive gaming is generally not feasible.

Despite this, many activities such as typical web browsing (and even watching YouTube videos, as long as you keep the resolution at 480p) are reasonably usable in Tor Browser. However, there is another usability issue affecting Tor Browser: when you're accessing a website that hides its IP address using Tor (such a website is called an "onion service"), the address doesn't look like something you can remember (e.g. "https://kipr.org"), it instead looks like this:

https://odmmeotgcfx65l5hn6ejkaruvai222vs7o7tmtllszqk5xbysola.onion

This insanely unmemorable and unusable address is structured so that Tor Browser can read the address and determine how to construct the correct chain of relays that can reach the website. There have been a few workarounds for this, such as:

- Use a centralized list of onion services, e.g. a wiki that lists all the known onion services and their names.

- Use a local list of onion services, e.g. bookmarks.

Both of these approaches introduce their own problems. Using a centralized list of onion services means that whoever runs that list can redirect you to an impostor website. And using a local list of bookmarks is unhelpful if a friend tells you to go to an onion service that you've never visited before.

For many years, it looked like this tradeoff was unsolvable. In fact, there was a formalization of this problem, called Zooko's Triangle [4], which stated "Global; Decentralized; Human-Meaningful: Choose Two." There was even a math proof by Leslie Lamport, going all the way back to the 1970's, that purported to prove the impossibility of this class of problems.

6 Namecoin

As it turns out, the class of problems that Lamport proved unsolvable, and which includes Zooko's Triangle, also includes the problem of producing a decentralized digital currency. This problem was solved in 2009 by Satoshi Nakamoto's invention of Bitcoin [5], by exploiting a loophole in Lamport's impossibility proof. By 2010, Bitcoin community members had realized that the same loophole could be used to solve Zooko's Triangle. In 2011, this was implemented as Namecoin [6], a modified version of Bitcoin that allows registering global, human-meaningful names and attaching short pieces of data to them. (Unlike Bitcoin, Namecoin is not designed to be used as a currency.)

Namecoin was originally intended to be used only for attaching IP addresses to names, but the community quickly realized that one could attach a Tor onion service address to a name, thus allowing Tor Browser users to access onion services via addresses like "https://kipr.bit/", without any of the drawbacks covered in Section 5.

I joined the Namecoin team in 2013, and onion service use cases were one of my prime focuses.

7 Tor Browser and Namecoin: Collaboration and Deployment

Namecoin has collaborated with the Tor Browser Team for a while on various areas of common interest, but things really kicked off in 2018 when I happened to notice a Tor Browser developer (Arthur Edelstein, now a privacy engineer at Brave) expressing interest in Namecoin on Twitter. I reached out, and we began discussing what might be needed to bundle Namecoin with Tor Browser. The requirements Arthur outlined were pretty steep – but this was to be expected. Tor Browser is the top privacy-focused browser in the world; they have very high standards. To make a long story short, we succeeded at meeting those requirements, and in December 2019, I gave a talk at the 36C3 hacker conference [7] announcing that experimental Namecoin support was now included in the Linux Nightly version of Tor Browser (Nightly is a special version of Tor Browser that includes all the latest features for adventurous users to experiment with, with the understanding that it will be less stable and more buggy than the usual release). The details of how we met those requirements are a fascinating story in their own right, and I'd highly recommend watching my 36C3 presentation if you're interested in the details.

8 Botball Skill Set Applied to Namecoin Development

The skill set I gained in Botball has been invaluable in working on Namecoin. My GCER 2017 paper, "Making HTTPS and Anonymity Networks Slightly More Secure" [1], discusses several of these, such as:

- Hacking Botball controllers is excellent practice for reverse-engineering.
- Double Elimination strategy is excellent practice for questioning security assumptions.
- The KISS Principle is excellent practice for reducing attack surface.
- International GCER paper collaboration is excellent practice for international software projects like Namecoin.

I'd highly recommend reading my 2017 paper if you're curious about these aspects. But, there are other aspects that I didn't cover in 2017, that I think are also great examples. In the below sections, I'll cover some of them.

8.1 Parts Lists Limits

It's hard to find a Botballer who hasn't tangled with the limits of the parts list. Whether it's because the parts list simply didn't include enough parts to build the thing they wanted, or being disqualified from a Double Elimination round because they used an illegal part, or memorizing the parts list in order to successfully get an opponent disqualified, the limits of the parts list are an integral part of Botball. Many Botballers who later move on to university-level KIPR-operated competitions like KIPR Open and KIPR Aerial are relieved by the lack of such constraints. While I agree that the lack of such constraints in the latter competitions can enhance creativity (*cough cough Team SNARC's "Rainman" series of KIPR Open bots that used a jumbo umbrella*), there is a substantial educational value in being able to work within arbitrary design constraints that limit the complexity you can deploy.

The most critical design constraint that Arthur from Tor imposed on Namecoin integration was that the total download size increase that resulted from adding Namecoin had to be under 3 MB. Given that our initial prototype added 40 MB to the download size, this was a serious constraint that took a lot of creativity to overcome. The constraint is entirely understandable from Tor's point of view: many Tor users have severe bandwidth constraints (especially users in the Global South). However, the fact that it's a reasonable constraint didn't make it any easier to comply with. We did eventually succeed at meeting this constraint, and I think one reason I was able to do this (and didn't just ragequit) was that I had a lot of experience from Botball in dealing with parts list limits, which are qualitatively similar.

In Botball, one of the deceptive aspects of the parts list is that even if your first choice for how to engineer something ends up getting excluded by parts list requirements, there's usually an alternative way to solve your problem that uses different parts, or uses the same parts more efficiently (perhaps with some tradeoffs involving programming effort). I observed similar deceptive aspects in trying to shrink Namecoin down to the 3 MB requirement that Arthur imposed. The parts list is an excellent aspect of Botball that provides real-world experience for software engineering constraints such as download size.

8.2 Documentation and GCER Paper/Presentations

Documentation seems to be a polarizing aspect of Botball. Many Botballers see it as pointless, while other Botballers genuinely enjoy writing excellent documentation. From my perspective, writing documentation for my code in Botball gave me the skill set I needed to comply with requirements from funders. The bulk of Namecoin's recent funding comes from NLnet Foundation [8], a Dutch nonprofit that is in turn funded by the European Commission and the Netherlands Ministry of Economic Affairs and Climate Policy. NLnet tends to dislike excessive paperwork or meetings, so they don't require us to constantly submit reports to them. However, they do need to know what we're doing. They achieve a balance by simply asking us to write regular public blogposts about what code we've written and how it works. This allows the general public to know what we're doing, and NLnet can read the blog just like any other member of the public can, which enables NLnet to know how things are going. The documentation I wrote in Botball bears an uncanny resemblance to the blogposts I wrote to comply with this NLnet requirement.

GCER papers also were great practice for this, and additionally, the practice I got giving technical presentations at GCER was invaluable for giving conference talks when representing Namecoin. I've given talks on Namecoin at various venues, such as the Internet Archive [9] (well-known for the Wayback Machine), ICANN [10] (they run Internet infrastructure such as the DNS root zone), and C3 (the largest hacker conference in Europe). I frequently get complimented on the quality of presentations I give on Namecoin, and this is definitely not a natural skill for me (I have Asperger Syndrome; public speaking is something I had to learn). The dominant reason I picked up this skill set was from giving GCER presentations. Most programmers don't gain any experience presenting at conferences until post-high-school – Botball's track record of preparing middle-school and high-school students to give technical presentations is phenomenal, and should be commended.

8.3 Open-Source Development

Botball tends to encourage open-source development. For example, many Botball teams (including Norman Advanced) routinely release open-source code at GCER, and KIPR themselves often release firmware source code so that adventurous teams who don't mind voiding the warranty can customize the controllers to suit their needs. As a result, it's common for Botball teams to fork KIPR's code and contribute changes back to KIPR, or forking other Botball teams' code and contributing those changes back. Botball teams routinely use tools like Git [11] to manage source code in a way that is conducive to open-source development.

At Namecoin, all our code is open-source and is developed in the open, because this is the only way to ensure that the code is secure. If your code's security is dependent on your adversary not knowing how it works, you don't have actual security. My first time using Git (and its predecessors, SVN and CVS) was in the context of Botball, and this experience made it a lot easier for me to start contributing to open-source projects like Namecoin in 2013. There aren't a lot of middle-school and high-school students using open-source development workflows; Botball has done an excellent job of fostering this kind of environment.

8.4 Collaboration with Nominal Competitors

While Botball is nominally a competition, everyone in Botball generally understands that the competition is friendly, and there's nothing wrong with swapping ideas (or code, or spare parts) with other teams. Norman Advanced has been known to donate its Time Out Card to an opponent, solely because it seemed like the right thing to do, and they would rather lose in a high-quality match instead of win because their opponent's bot broke during setup. Similarly, Norman Advanced routinely co-authors GCER papers with other teams, among many other forms of cross-team collaboration.

Namecoin also has competitors in the space, and we approach this competition much like Botball teams do. For example, two of our competitors are Monero [12] and Handshake [13], and I routinely swap ideas with the teams behind those projects. Our three projects all understand that our primary goal is to make the Internet more secure, not to win battles with each other. As such, our projects are happy to learn from each other, thus making all of our projects better. The Monero team routinely invites us to conferences (my 36C3 talk about Tor Browser integration was hosted on a stage run by the Monero team), and the Handshake team even donated an airdrop of a million dollars to fund Namecoin development.

9 Conclusion

Botball is a top-notch educational program that prepared me for my current work with Namecoin. While most discussion at GCER focuses on robotics ("Robotics" *is* in the name of the conference, after all!), I think it's important from an educational research perspective to analyze the non-robotics-specific skills that Botball imparts in its students. It is my hope that GCER will include more coverage in the future of the non-robotics projects that alumni are working on. After all, while robotics is awesome, it's still only a small fraction of the software development industry. Recognizing that Botball provides critical educational benefits across the programming spectrum is, in my opinion, key to making sure Botball gets the attention it rightly deserves.

And finally, a career plug to Botball students: if you think privacy should be a human right, and you find security and cryptography to be a fascinating topic, we'd love to have you at Namecoin. Blockchain projects in particular seem to be a hot topic in industry, and any kind of open-source development tends to look great on a resume (especially if you did it before graduating high school). If you might be interested in working with us, email me (jeremy@namecoin.org). If you're not confident in your knowledge of security topics, that's not a problem – as documented in this paper and my previous paper from GCER 2017, being in Botball means you probably already have a pretty good skill set here, even if you don't realize it.

10 References

[1] Jeremy Rand. Making HTTPS and Anonymity Networks Slightly More Secure (Or: How I'm Using My Botball Skill Set in the Privacy/Security Field). GCER 2017. https://ynqrnzvuotdgvqj2g73f5tyj4rcvwutdemu23jgadawihbkc3lxkbwid.onion/pdf/gcer/2017/Namecoin_GCER_2017.pdf

[2] Freedom Outpost. Former CIA / NSA Director Michael Hayden: “We Kill People Based on Metadata”. https://investortimes.com/freedomoutpost/former-cia-nsa-director-michael-hayden-kill-people-based-on-metadata/

[3] The Tor Project. https://www.torproject.org/

[4] Zooko Wilcox. Names: Distributed, Secure, Human-Readable: Choose Two. https://web.archive.org/web/20011020191610/http://zooko.com/distnames.html

[5] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf

[6] Vincent Durham, et al. Namecoin. https://www.namecoin.org/

[7] Jeremy Rand. 36C3 Summary. https://www.namecoin.org/2020/01/11/36c3-summary.html

[8] NLnet Foundation. https://nlnet.nl/

[9] Internet Archive. https://www.archive.org/

[10] Internet Corporation for Assigned Names and Numbers. https://www.icann.org/

[11] Linus Torvalds, et al. Git. https://git-scm.com/

[12] Monero Developers. Monero – Secure, Private, Untraceable. https://www.getmonero.org/

[13] Handshake Developers. Decentralized naming and certificate authority. https://handshake.org/