Making HTTPS and Anonymity Networks Slightly More Secure
(Or: How I'm Using My Botball Skill Set in the Privacy/Security Field)
Jeremy Rand
The Namecoin Project / Norman Advanced Robotics Alumni / Team SNARC Alumni
jeremy@namecoin.org

Making HTTPS and Anonymity Networks Slightly More Secure (Or: How I'm Using My Botball Skill Set in the Privacy/Security Field)

1 Introduction

The topic of online privacy has been receiving quite a lot of attention in the past few years. Much of the recent attention has focused on the advice "Use HTTPS!" and "Use an anonymity network like Tor!" Unfortunately, while HTTPS and Tor are excellent, they're not perfect, and some of their security flaws were, until recently, believed to be mathematically impossible to fix. It turns out, however, that Bitcoin technology can fix those flaws that were believed to be unfixable. I'm a developer of Namecoin, which implements those fixes. And the skill set I'm using at Namecoin is pretty much exactly the skill set I learned as a Botball student.

In this paper, I'll cover what Namecoin is, why it's useful for online privacy, and how competing in Botball prepared me for this field of work.

2 Who Are You?

I'm a former Botballer from Norman Advanced Robotics in Norman, OK (class of 2011). I used to lead Team SNARC, which competed in KIPR Open and KIPR Aerial between 2011 and 2015 (we won a few awards). I also mentored three middle-school Botball teams in Norman (Alcott, Whittier Boys, and Whittier Girls) between 2011 and 2015. In Botball, I tended to specialize in hacking Botball controllers; my various GCER papers included hacking the XBC [1], CBC [2], Link [3], AR.Drone [4], and Create [5]. Outside of robotics, I used to do console game hacking, including hacking an offline 2-player GameCube game (Sonic Adventure 2: Battle) to play online with unlimited simultaneous players, and hacking another GameCube game (Starfox Assault) to play in virtual reality with the Oculus Rift and Razer Hydra. These days, I tend to spend most of my time on human rights technologies such as Namecoin [6], YaCy [7], and Replicant [8].

3 DNS

DNS can be thought of as the "phone book" of the Internet. When you try to visit "www.botball.org" (called a *domain name*), your computer doesn't know how to route traffic to a domain name, so it uses the DNS (Domain Name System) to look up the IP address associated with that domain name, which in this case is "66.33.207.219".

DNS is an example of a *naming system*, which converts short names into arbitrary data. There are lots of features that a naming system could have that are desirable, but three of them are:

1. Global. Looking up a particular name will result in the same data, regardless of where you are when you do the lookup.
2. Human-meaningful. The names are something that a human can understand, choose, and remember.
3. Decentralized. The naming system does not rely on a centralized, trusted authority in order to get the correct data for a name.

Zooko Wilcox formalized these three criteria as Zooko's Triangle [9], and he conjectured "choose two"; i.e. that he believed it was impossible to achieve all three criteria simultaneously. DNS is global and human-meaningful, but not decentralized. Other naming systems exist that choose a different set of two out of three.

4 TLS

You've probably heard that HTTPS is better than HTTP, and that the S stands for "secure". What HTTPS is doing under the hood to make your connection secure, is that it combines HTTP with a protocol called TLS [10]. TLS is designed to use encryption to prevent anyone but the intended recipient from reading the HTTP traffic. TLS uses *asymmetric encryption*; this means that there are two numbers (called *keys*) that a website has: a *public key* (which everyone knows) and a *secret key* (which only the website's server knows). Asymmetric encryption has two important properties. First, if you know the secret key, you can easily calculate the public key, but not vice versa. And second, if you possess the public key for an HTTPS website, you can encrypt data in such a way that only the holder of the secret key can decrypt it.

Unfortunately, there's a problem here: you probably don't know what the correct public key is for the websites you visit. So what stops an attacker from giving you the *wrong* public key for a website, and thus tricking you into encrypting the data so that *the attacker* is the one who can decrypt it? In fact, what stops an attacker from doing this, and then *re-encrypting* that data using the website's correct public key, and passing it on to the website so that both you and the website don't see anything obviously wrong? This is called a *Man-in-the-Middle attack*, abbreviated *MITM*.

The important thing here is that in order to prevent MITM attacks, you need some way of figuring out what the correct public key is for every HTTPS website that you visit. How can we do that?

5 Certificate Authorities

The way that the Internet community decided to prevent MITM attacks, is by using the equivalent of ID cards for proving authenticity of website public keys. Corporations called Certificate Authorities (CA's) issue *certificates* to websites. The certificates cryptographically prove that a particular CA believes that a specific public key is valid for a particular domain name. When you visit an HTTPS website, the website's server shows your web browser its certificate, and your web browser verifies a few things:

1. The certificate is issued to the same domain name as the one your browser is trying to connect to.

2. The web server possesses the secret key that corresponds to the public key listed in the certificate.
3. The certificate is issued by one of the CA's that your browser trusts.

If any of these criteria fail to verify, your web browser concludes that a MITM attack might be happening, and it cancels the connection instead of sending any data.

There are over 1000 CA's that your browser trusts [11]. You might see a problem here.

6 Are the CA's Trustworthy?

If any of the thousands of CA's that your browser trusts is malicious, extorted, or just tricked, they could issue a fraudulent certificate. This would allow a MITM attack to go undetected. This actually happens in the real world from time to time.

The most well-known case is from 2011: an attacker (who may have been an Iranian intelligence agency) compromised the CA DigiNotar [12], and issued themselves fake certificates allowing them to impersonate hundreds of targets. These targets included the CIA, MI6, Facebook, Microsoft, Skype, Twitter, WordPress, and Mozilla [13]. These fraudulent certificates were later found deployed on an Iranian server [12], presumably being used to wiretap Iranian Internet users.

And no one even noticed that this had happened for over a month [12].

There have been notable screw-ups by CA's that didn't require getting their systems cracked. In 2016, the CA WoSign issued a certificate for github.com to someone who only proved that they had a user account on GitHub [14]. Also in 2016, the CA Comodo issued a certificate to the wrong person because the CA used OCR image-scanning software to read a picture of the website owner's email address, and the OCR algorithm misread the email address [15].

Some CA's may also be intentionally malicious. The U.S. company Blue Coat set up their own CA in 2016; Blue Coat is well-known for creating mass surveillance systems which were covertly sold to the Syrian government. (We only know that Blue Coat's devices were deployed in Syria because the hacker group Telecomix leaked the log files from those devices in 2011 [17].) And after the Heartbleed emergency required many certificates to be quickly revoked and replaced, the CA StartCom held all their users hostage, demanding that exorbitant fees be paid in order to disable certificates whose private keys had been leaked [18].

Unfortunately, even in cases where we have strong reason to suspect individual CA's of wrongdoing, many CA's have attained "Too Big To Fail" status. In the same way that U.S. banks held the world economy hostage in 2008 in order to receive government bailouts, many CA's are able to assert that if they are shut down, all the websites to whom they issued a certificate will stop working, which would be harmful to the economy. (This problem is so pervasive that it became the subject of a satirical Mozilla bug report asking Mozilla to add the CA for "Honest Achmed's Used Cars and Certificates", whose unique selling point was being honest about intending to become Too Big To Fail [19].)

This leads to a question: is there a way to verify certificates' trustworthiness without requiring CA's at all?

7 What about using DNS as a CA substitute?

In theory, you could make the DNS provide TLS public keys as well as IP addresses when you do a lookup. In fact, there's a standard for this called *DANE*. Unfortunately, while this is probably an improvement over the CA system, the primary effect is that it re-shuffles the set of trusted third parties rather than eliminating them. In particular, the DNS's centralized architecture means that the company from whom you registered your domain name, as well as the owner of the top-level domain (e.g. .com or .org) that you're using, as well as the root key (held by ICANN, the Internet Corporation for Assigned Names and Numbers), can all act as CA's under such a system.

Is this bad? Consider the popular URL shortener bit.ly [21]. .ly is the Libyan top-level domain. If the DNS were used for TLS trust today, then what happens to bit.ly, all the people who have created bit.ly shortened URL's, and all the people who click on such URL's, when, as seems plausible to occur soon, the terrorist group IS finishes taking over Libya and gets control of the TLS for all the .ly domain names? (Not that Ghaddafi's government was necessarily trustworthy either.)

Also consider that domain names don't usually change: bit.ly can't easily switch to a new top-level domain because they're worried that Libya might soon be controlled by IS. Switching CA's is easy by comparison.

8 Namecoin

This begs a question: can we have a DNS-like naming system that has the DNS's advantages of global, human-meaningful names, but is also decentralized and doesn't require trusting third parties?

Appamatto [22] and Aaron Swartz [23] observed that solving Zooko's Triangle seems to be very similar in nature to the problem that Bitcoin solves: the ability to come to a decentralized, global consensus on an ordered sequence of events. (In Bitcoin, the events are financial transactions.) Like Zooko's Triangle, decentralized global consensus was once believed to be an impossible problem, but Bitcoin is a solution. Appamatto and Swartz suggested that you could implement a naming system that meets all three of Zooko's Triangle's requirements similarly to Bitcoin, by making the events consist of registration of domain names and updating their data, instead of simply being financial transactions.

Vincent Durham implemented this concept as Namecoin. Namecoin domain names use the ".bit" top-level domain. Namecoin is designed to be easily interoperable with the DNS, which means that systems like DANE can easily be adapted to work with Namecoin. This provides a system of verifying what the correct TLS public key is for a given website without using a CA or the DNS. The only requirements are that the website register a .bit domain name with the Namecoin software, and that the user viewing the website have the Namecoin software installed so that they can do the lookups.

Since Namecoin was created in 2011, we've been favorably mentioned in an ICANN report [24], been an invited speaker at an ICANN meeting [25], been favorably mentioned by Internet Archive founder Brewster Kahle [26], been endorsed by the whistleblower and privacy advocate organization WikiLeaks [27] (Julian Assange also mentioned us to Google CEO Eric Schmidt

[28]), been utilized by the NSA-leaking group TheShadowBrokers [29], and received funding from the NLnet Foundation's Internet Hardening Fund (which is funded by the Netherlands Ministry of Economic Affairs) [30]. (A rather diverse set of fans.)

9 Anonymity Network Naming

TLS's encryption can protect the content of communication, but it doesn't hide *metadata*, i.e. who communicated with whom, and when. Unfortunately, metadata can't be ignored; often metadata is sufficient to land human-rights activists in prison even if the content of their communications was protected. Anonymity networks like Tor [31] solve this issue by bouncing communications between a sequence of relays, which obscures the origin and destination of communications.

Tor's anonymity is excellent, but a problem with Tor is that hosting a website behind Tor requires that users access the website using a ".onion" URL. .onion falls on the "global and decentralized, but not human-meaningful" end of Zooko's Triangle. For example, a typical .onion URL might look like this:

https://idnxcnkne4qt76tg.onion

If you think that's difficult to remember, note that Tor is currently doing a cryptography upgrade, so soon they'll look like this instead:

https://odmmeotgcfx65l5hn6ejkaruvai222vs7o7tmtllszqk5xbysola.onion

We're examining using Namecoin as a DNS-like system for anonymity networks like Tor. Conceptually, this would work by allowing a Namecoin domain name to return a .onion address in addition to an IP address or a TLS public key. Doing this safely would require Namecoin itself to be anonymous, which is a task that we're in the process of addressing.

10 Botball Skill Set: Reverse-Engineering

In Botball, students are often given controllers whose software is not completely documented, and whose functionality doesn't necessarily meet what the students would want to do. The standard workaround in Botball is to study how the software works, based on the documentation that is available as well as experimental methods, and eventually reverse-engineer the software enough to figure out how to get it to do what's needed. This was the basic formula for all of my Botball hacking papers.

It turns out that the TLS software built into standard web browsers is quite similar to this: there's no built-in way to make a web browser use Namecoin for TLS, and the web browser's internals aren't usually well-documented. As part of my work developing Namecoin's TLS support for Chromium on Windows, I ended up needing to reverse-engineer how the Windows CryptoAPI worked internally, so that I could fiddle with its list of trusted certificates. Doing this brought on a rush of nostalgia – I felt like I was hacking Botball controllers again. I am confident that the reason I succeeded at this task was the reverse-engineering experience I had gained years before with Botball.

11 Botball Skill Set: Questioning Assumptions about Adversaries

In Botball's Double-Elimination (DE) rounds, one of the most important deciding factors in who wins has nothing to do with how well-built or well-programmed a robot is. That deciding factor is the ability of a team to accurately guess what other teams will try to do. This can play a role in strategies that prevent the opponent from scoring, and it can also play a role in strategies that can score reliably even when the opponent is trying to prevent you from scoring.

This basically comes down to one skill: the ability to question your assumptions about adversaries, question your assumptions about how the adversaries will interact with you, and guess what incorrect assumptions your adversary made that you can take advantage of. The same is true in the computer security field; pretty much all of the work being done in computer security involves figuring out where people's assumptions can fail.

As one example, while I was developing Namecoin's TLS support for Chromium, I was able to get Chromium to recognize Namecoin certificates as valid, but I realized that it would be an incorrect assumption to think that this was sufficient. The flip side was that I needed to make sure that no CA-issued certificates for Namecoin domain names would be considered valid. (CA's aren't *supposed to* issue certificates for Namecoin domain names, but if we could trust CA's to behave properly, we wouldn't need Namecoin.) We don't want Namecoin to only be secure *until* a malicious CA issues a cert that claimed to be valid for a Namecoin domain name. I ended up using a feature in Chromium called *key pinning*, which prevents any CA from being accepted for a domain name unless the CA is in a whitelist. Unfortunately, an empty whitelist just gets ignored, and I didn't want to allow *any* CA's to be allowed. However, the CA whitelist is a list of public keys held by the CA's, so I could just list a public key that didn't correspond to any CA. (Technically these are public key *hashes*, not public keys, but you can think of them as public keys for this purpose.) But here's where another assumption has to be avoided: I could just make up a public key, but what would stop me from secretly possessing the secret key for that whitelisted public key, and using it to compromise the system (often referred to as a *back door*)? Telling people "trust me" isn't acceptable, since "trust me" is exactly the problem with CA's. However, it's possible to construct a public key for which it is provable that no one knows the corresponding secret key. This technique is called a *Nothing Up My Sleeve* (NUMS) technique [32]; basically it involves taking a well-known mathematical constant (e.g. *pi*) and using it as a public key. This demonstrates that the public key was generated in a way that didn't involve possession of the secret key, and that therefore that no one is likely to possess the secret key – no need to trust us, just compare the public key to *pi*.

Another example of questioning assumptions is that there exists a Firefox extension that can block connections that have invalid certificates according to DANE [33]. I considered adapting it to use Namecoin, but upon auditing its source code, I noticed that it wasn't actually checking the certificate until the server had sent a web page to the browser. This means that the initial request from the browser to the server was allowed to complete before the certificate check happened. Among other things, this meant that sensitive data such as login cookies could be transmitted and stolen before the connection was blocked. I checked this experimentally by doing a MITM attack against myself with that Firefox extension installed, and sure enough, even though the connection was blocked by the extension, it was too late – my MITM attack had recovered the login cookies, which would be sufficient to commit identity theft. When I asked the developers of that Firefox extension about this issue, it turned out that their design made the

assumption that the only purpose of verifying TLS certificates was to block the browser from displaying a website that failed to verify; they didn't consider the need to prevent a MITM attacker from stealing the data transmitted to and from the website. Needless to say, Namecoin didn't use that Firefox extension's code.

12 Botball Skill Set: Minimizing Attack Surface

In Botball, a common theme is the KISS Principle. This is based on the observation that more complex systems are more likely to go horribly wrong, simply because there are more components that can go wrong. In computer security, we have a similar concept: *attack surface*. Computer security engineers tend to try to minimize attack surface in several ways: reducing the complexity of the system overall, isolating the most high-risk parts of the system from the parts of the system that are most likely to be exposed to attacks, and avoiding techniques that are historically known to fail catastrophically.

Reducing complexity is, of course, common in Botball, and is also a common goal in computer security. For Namecoin, we initially used a technique known as an *intercepting proxy* to make browsers work with Namecoin for TLS. An intercepting proxy is basically a "friendly MITM attack", where the MITM is done by a proxy software running on the user's own computer. The proxy can implement whatever rules it wants for validating the server's TLS certificate, and if it passes validation, the proxy can make the certificate look valid to the user's web browser. However, we abandoned this approach because it was way too complex – it requires the proxy to re-implement all of TLS (a complicated protocol), and if it makes a mistake, the web browser will have no way of noticing. Our current approach has substantially lower attack surface.

Security by isolation happens in Botball too. For example, the use of blocking robots to prevent opponents from getting near the robot that's scoring points can be thought of as a method of reducing attack surface via isolation: the blocking robot is likely to be "attacked" but is expendable, whereas the robot that is scoring points is made less likely to be attacked. In computer security, isolation might be implemented by keeping sensitive code/data and network-facing code in separate processes (or even separate VM's [34]), so that a compromise of the network-facing code is less likely to metastasize to the sensitive code/data.

Avoidance of known past failure modes is also important. In Botball, teams build up an institutional memory of what worked well and what didn't, and they also are likely to learn what strategies spectacularly backfired for other teams in tournaments. In computer security, one of the most common failure modes is memory safety vulnerabilities, which disproportionately affect C code. As a result, most of the competent security engineers are favoring other languages like Go and Rust, which avoid the memory safety issues that have, over time, been shown to be a persistent problem in C code [35].

13 Botball Skill Set: International Collaboration

It's not unheard of for Botball teams in different states, or different countries, to collaborate. When I was at Norman Advanced in Oklahoma, we regularly collaborated with Nease High School in Florida. More recently, I'm aware that Norman Advanced has collaborated with at least one team in Austria.

Open-source projects like Namecoin take this a step further. The Namecoin team itself is

scattered across countries: we have developers in the U.S. (including Oklahoma, Texas, Washington state, and Connecticut), Switzerland, Germany, the U.K., Sweden, and Canada, as well as former developers in France and Russia. Several of our developers operate under pseudonyms, and some don't disclose anything about where they're based. Development is coordinated entirely via the Internet. We also collaborate with related projects, such as Monero [36] (an anonymity-focused cryptocurrency that we have a particular interest in for the purpose of making Namecoin safer to use with Tor). Most of Monero's developers are pseudonymous; our primary contact at Monero is in South Africa. (I eventually met him in person at a conference in London in March 2017 where we were both speaking, after having worked with him online for a couple of years). Botball was excellent practice at international project collaboration.

14 Conclusion

My experience in Botball has been an excellent preparation for my current work with Namecoin, and the Botball program deserves significant credit for accurately representing the challenges that show up in "the real world".

To any Botball students reading this paper: if you find cryptography or security to be interesting topics, and you like the idea of making the world a better place for human rights such as privacy and free speech, we'd love to have your help at Namecoin. Working on open-source software development is a great way to boost your resume and gain real-world experience (and making a positive difference in the world is good too, of course). Send me an email (jeremy@namecoin.org) if you're interested, or check out https://www.namecoin.org/ for more information. Even if you don't think security is something you're particularly good at, you might be surprised – when I was at Norman Advanced Robotics showing a Botball special awards judge how I had hacked the iRobot Create, he asked if I had considered cryptography/security as a career, and I said that I found the topic interesting but didn't consider myself good enough at it to pursue a career in it. I guess the joke was on me.

15 References

[1] Jeremy Rand and Fahrzin Hemmati. *Hacking the XBC Firmware: Programming the XBC in Standard C++*. GCER 2008.

[2] Jeremy Rand, Matt Thompson, and Braden McDorman. *Hacking the CBC Botball Controller: Because It Wouldn't Be a Botball Controller if It Couldn't Be Hacked*. GCER 2009.

[3] Jeremy Rand. *Hacking the KIPR Link: Using the Pixy Low-Latency USB Color Camera*. GCER 2015.

[4] Jeremy Rand. *AR.Pwn: Hacking the Parrot AR.Drone*. GCER 2013.

[5] Jeremy Rand. *Create Open Interface Scripts: Using the Create Without an XBC*. GCER 2008.

[6] Namecoin Developers. *Namecoin*. https://www.namecoin.org/

[7] YaCy Developers. *YaCy Search Engine*. https://github.com/yacy/

[8] Replicant Developers. *Replicant*. https://www.replicant.us/

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

[10] Wikipedia Contributors. *Transport Layer Security*. https://en.wikipedia.org/wiki/Transport_Layer_Security

[11] Jesse and Peter Eckersley. *Is the SSLiverse a safe place?* https://media.ccc.de/v/27c3-4121-en-is_the_ssliverse_a_safe_place

[12] Chester Wisniewski. *Falsely issued Google SSL certificate in the wild for more than 5 weeks*. https://nakedsecurity.sophos.com/2011/08/29/falsely-issued-google-ssl-certificate-in-the-wild-for-more-than-5-weeks/

[13] Chester Wisniewski. *SSL certificate debacle includes CIA, MI6, Mossad and Tor*. https://nakedsecurity.sophos.com/2011/09/05/ssl-certificate-debacle-includes-cia-mi6-mossad-and-tor/

[14] Mike Masnick. *Certificate Authority Gave Out Certs For GitHub To Someone Who Just Had A GitHub Account*. https://www.techdirt.com/articles/20160825/12181835347/certificate-authority-gave-out-certs-github-to-someone-who-just-had-github-account.shtml

[15] steffen. *Comodo: CA Comodo used broken OCR and issued certificates to the wrong people*. https://bugzilla.mozilla.org/show_bug.cgi?id=1311713

[16] Filippo Valsorda. *BlueCoat now has a CA signed by Symantec.…* https://twitter.com/filosottile/status/735940720931012608?lang=en

[17] Andy Greenberg. *Meet Telecomix, The Hackers Bent On Exposing Those Who Censor And Surveil The Internet*. https://www.forbes.com/sites/andygreenberg/2011/12/26/meet-telecomix-the-hackers-bent-on-exposing-those-who-censor-and-surveil-the-internet/

[18] Mike Masnick. *Shameful Security: StartCom Charges People To Revoke SSL Certs Vulnerable To Heartbleed*. https://www.techdirt.com/articles/20140409/11442426859/shameful-security-startcom-charges-people-to-revoke-ssl-certs-vulnerable-to-heartbleed.shtml

[19] Honest Achmed. *Add Honest Achmed's root certificate*. https://bugzilla.mozilla.org/show_bug.cgi?id=647959

[20] Wikipedia Contributors. *DNS-based Authentication of Named Entities*. https://en.wikipedia.org/wiki/DNS-based_Authentication_of_Named_Entities

[21] Bitly. *Bitly | URL Shortener and Link Management Platform*. https://bitly.com/

[22] Appamatto. *BitDNS and Generalizing Bitcoin*. https://bitcointalk.org/index.php?topic=1790.0

[23] Aaron Swartz. *Squaring the Triangle: Secure, Decentralized, Human-Readable Names*. https://web.archive.org/web/20170424134548/http://www.aaronsw.com/weblog/squarezooko

[24] Internet Corporation for Assigned Names and Numbers. *Identifier Technology Innovation Report*. https://www.icann.org/en/system/files/files/iti-report-15may14-en.pdf

[25] Jeremy Rand. *ICANN58 Summary*. https://www.namecoin.org/2017/04/17/icann-58-summary.html

[26] Brewster Kahle. *Locking the Web Open – a Call for a New, Decentralized Web*.

https://archive.org/details/DWebSummit2016_Introduction_Brewster_Kahle

[27] WikiLeaks. *Namecoin and Bitcoin will be revolutionary….* https://twitter.com/wikileaks/status/78906603948093440

[28] WikiLeaks. *Transcript of secret meeting between Julian Assange and Google CEO Eric Schmidt*. https://wikileaks.org/Transcript-Meeting-Assange-Schmidt.html#737

[29] TheShadowBrokers. *TheShadowBrokers*. https://twitter.com/shadowbrokerss

[30] Jeremy Rand. *Namecoin Receives Funding from NLnet Foundation's Internet Hardening Fund*. https://www.namecoin.org/2017/05/19/funding-nlnet.html

[31] The Tor Project. *Tor Project: Anonymity Online*. https://www.torproject.org/

[32] Wikipedia Contributors. *Nothing up my sleeve number*. https://en.wikipedia.org/wiki/Nothing_up_my_sleeve_number

[33] cz.nic. *DNSSEC/TLSA Validator*. https://www.dnssec-validator.cz/

[34] The Qubes OS Project and others. *Qubes OS: A reasonably secure operating system*. https://www.qubes-os.org/

[35] Tony Arcieri. *It's time for a memory safety intervention*. https://tonyarcieri.com/it-s-time-for-a-memory-safety-intervention

[36] Monero Developers. *Monero – secure, private, untraceable*. https://getmonero.org/home