I Won Awards for Hacking Botball Controllers; Now I’m Protecting the Anonymity of At-Risk Internet Users
(Or: How Botball Prepared Me for a Career in Security/Privacy Research)
Jeremy Rand
The Namecoin Project / Norman Advanced Robotics Alumni / Team SNARC Alumni
jeremy@namecoin.org / jeremyrand@danwin1210.de

I Won Awards for Hacking Botball Controllers; Now I’m Protecting the Anonymity of At-Risk Internet Users (Or: How Botball Prepared Me for a Career in Security/Privacy Research)

1 Introduction

When a special awards judge at the 2009 Oklahoma Botball regional tournament told me I should think about a career in computer security, I didn’t really take him seriously at first. Sure, the hacking adventures I had just told him about (I had rigged the iRobot Create chassis from the Botball kit to drive around, using sensors, without a controller) was a cool achievement, but (as I told him) I had viewed hacking as more of a hobby, not really something I’d be good enough at to do professionally. He told me I might want to reconsider that assessment.

16 years later, I’m working in the privacy/security/cryptography sector, and it’s pretty clear that he was right. Botball isn’t just a game or a competitive sport, it’s an environment where students prepare for the real world of STEM – even if they aren’t consciously aware of it themselves. This paper is a case study of several privacy-focused research projects I’m involved with at Namecoin, and the parallels between the skill set I’m using now and what I picked up in Botball. I think post-Botball case studies are an underappreciated part of GCER – nothing beats real-world examples when evaluating how Botball enables its participants to succeed in the future.

This paper can be considered as “Part 3” of my series of recent GCER papers (“Making HTTPS and Anonymity Networks Slightly More Secure” [1] and “From Hacking Botball Controllers to Having My Code in the Number One Privacy Web Browser” [2]), but each paper in this series can be read on its own.

2 My Background in Botball

I discovered Botball in Fall 2002 while a 7th-grader at Whittier Middle School in Norman, OK; it didn’t take much convincing for me to join Whittier’s new team, and I was Whittier’s lead programmer for the 2003 and 2004 seasons. I stuck with Botball in high school, as a senior programmer at Norman Advanced Robotics (2005-2011). After high school, I founded and led Team SNARC, which competed in KIPR Open and KIPR Aerial for 4 years. At both Norman Advanced and Team SNARC, I quickly developed a specialty in hacking the Botball controllers; I published GCER hacking papers pertaining to the XBC, CBC, Link, AR.Drone, and Create, and several of these hacking adventures won Judges’ Choice awards.

Around the same time that I founded Team SNARC as a university freshman, KIPR gave me a heads up that Alcott Middle School was in need of a mentor. I offered to help out, initially assuming that I'd only be involved for a few months, but mentoring Botball turned out to be sufficiently fun that I stuck around for the entire 4 years that I was also leading Team SNARC. Outside of team activities, I was a founding member of the Botball Youth Advisory Council, and I also interned at KIPR for a few summers.

In 2013, after what I initially thought would be a month-long for-fun side project implementing a feature for a bounty for The Namecoin Project (I won 2 bitcoins as a bounty… not much at the time, but you can do the math to figure out how much that is as present exchange rates), I was asked to stick around as a Namecoin developer. I did so, and became a full-time Namecoin developer as soon as I finished my master's degree in 2018.

3 Background: Namecoin

Namecoin [3] was founded in 2011 by Vincent Durham, as a spin-off of Bitcoin [4]. Whereas Bitcoin focused on decentralizing the banking system, Namecoin was designed to apply similar techniques to decentralizing another piece of infrastructure: the DNS. The DNS is essentially the Internet's "phone book". When you type "`www.kipr.org`" into a web browser, that string of text (called a domain name) isn't helpful in routing your request to KIPR's website. The Internet uses numerical IP addresses to route data to servers, not human-readable strings. Under the hood, your computer solves this by asking the DNS to look up the IP address that corresponds to "`www.kipr.org`", which (at the time of writing) is "`66.33.202.71`".

Bitcoin's unique design is relevant for implementing a DNS-like system because Bitcoin solves the following problem: how do we come to a global consensus on the ordering of a sequence of events, in a decentralized manner? In Bitcoin, the events are financial transactions; in Namecoin, the events are registrations of domain names and updates of what IP address they point to. The ordering of events is equally critical in both systems: if you could change the order of who received money first, it would let you steal money; if you could change the order of who registered a domain name first, it would let you steal website addresses.

Although The Namecoin Project initially was focused on the "let's decentralize the DNS" use case, the project attracted quite a few skilled developers, and slowly morphed into a more generic privacy/security research group. We still do spend a lot of our time on DNS-related activities, but we also have a variety of side projects. In many cases, our side projects came into existence either because we needed a tool to do our DNS-related work and such a tool didn't exist, or because we had picked up some unique expertise while doing DNS-related work and we realized that this expertise enabled us to solve some other problem that would benefit the public.

4 Project Case Study 1: Encaya, TLS, and Tor

4.1 Intro to TLS

As you're probably aware, HTTPS is the secure version of HTTP, and websites whose URL begins with "`https://`" are more secure than those that begin with "`http://`". Under the hood, the S in HTTPS is a security layer called TLS. TLS is an example of *asymmetric encryption*; this means that the TLS protocol uses two different numbers (called *keys*) associated with a given website. The website has a *public key* (which, as the name implies, is public knowledge) and a *secret key* (which, as the name implies, is only known by the owner of the website). These two keys are mathematically linked in a way that ensures an interesting attribute: anyone who possesses the public key can encrypt data in such a way that only the holder of the secret key can decrypt it. Generally, authentication of which public key corresponds to which website is handled by corporations called *certificate authorities* (CA's). As you might guess, TLS isn't very useful if the CA's turn evil – the encryption could easily be intercepted by tricking your computer into encrypting traffic to a public key whose secret key is held by an attacker.

4.2 Intro to Tor

Tor [5] is an anonymity network that bounces communications through a series of volunteer-run relays to ensure that users can talk to each other without knowing each each other's identity or location. The communications are encrypted multiple times in a layered fashion, with each relay peeling back one layer of encryption before passing the traffic to the next relay in the chain. The layered encryption ensures that each relay only knows about the two hops that are adjacent to it in the chain. The first relay knows who you are, but has no idea whom you're talking to – it only knows what relay you've picked as the second hop. The final relay knows whom you're talking to, but has no idea who you are – it only knows the second-to-last relay that you picked for the relay chain.

Tor is an excellent tool for privacy-conscious users. It has successfully kept whistleblowers safe, and is also beneficial for average users who want to avoid having their privacy violated by targeted ads, or who live in countries where access to accurate news is censored by their government. Websites that are hosted anonymously over Tor are called *onion services*, and can be easily recognized by their unique URL style. For example, the onion service of The Tor Project's website can be found at:

http://2gzyxa5ihm7nsggfxnu52rck2vv4rvmdlkiu3zzui5du4xyclen53wid.onion/

You may notice a few curious attributes of this URL. First off, it ends with "`.onion`", not something more common like "`.org`". Second, the rest of the URL is random letters and numbers, not something human-readable. Third, the URL uses HTTP, not HTTPS.

The "`.onion`" suffix is simply there so that your web browser knows it's an onion service. The randomness of the rest of the URL has a specific purpose: onion services use asymmetric encryption just like TLS, and the random data is actually a public key! This means that if you're accessing an onion service, you already know what public key you need to encrypt to; there's no need to ask a

corporation like a CA who might lie to you. The lack of HTTPS is, in theory, because onion services are already encrypted, and TLS would be redundant.

4.3 TLS + Tor?

However, theory doesn't always match reality. In the real world, TLS does grant some useful security protections that onion services don't. There is thus an advantage to combining onion services with TLS. Unfortunately, this is harder than it sounds. The easy way would be to convince CA's to support onion services – then everything should "just work" the same way TLS normally does. Unfortunately, this would mean that we're having to trust the CA's.

A more clever approach occurred to me: onion services have their own public key built into the URL. Could we rig web browsers to use that public key instead of asking a CA what public key should be used?

At Namecoin, I had built up a lot of expertise in rigging web browsers to authenticate public keys via nonstandard mechanisms – Namecoin uses a public key that's embedded next to the IP address in the Namecoin blockchain. The result of this expertise was a tool I had created, called *Encaya*. Encaya is essentially a compatibility shim that teaches standard web browsers how to discover TLS public keys via nonstandard mechanisms.

As I examined the task of making Encaya work with onion services, a problem quickly became apparent. Onion services use a different public key type than TLS. It's not possible to convert between these key types, but it *is* possible to make an onion secret key sign a TLS public key. If you have the signature, then you can confirm that the TLS key is authorized by the onion key.

Encaya works by hooking into a standard interface that web browsers use to look up TLS public keys: the web browser sends Encaya the URL of the website in question, and some related data, including a serial number for that website that was purportedly assigned by a CA. I realized that we could do something crazy here: we could disguise the onion key's signature of the TLS key as a serial number! This way, Encaya would be able to extract the onion public key from the URL, extract the signature hidden inside the serial number, and verify the signature against the onion public key. If it matched, then Encaya would know which TLS public key to authorize for that onion service.

Sure enough, I got it to work. I presented a prototype of this work at the Tor Developer Meeting in May 2024, and subsequently presented it at the GPN22 [6] and 38C3 [7] hacker conferences later that year. Namecoin now has funding from NLnet Foundation [8] and Power Up Privacy [9] to move this closer to deployment.

5 Project Case Study 2: SocksTrace and Proxy Leaks

Tor is excellent for anonymity, but it can't help you if you use it wrong. Making an application such as Firefox use Tor involves configuring the application's proxy settings. (You might have encountered proxy settings when configuring your personal laptop to use a school or corporate network.) If the application sends anything without going through Tor's proxy (called a *proxy leak*), your anonymity is gone.

At Namecoin, we develop software that needs to route its traffic through Tor, and we wanted to make sure our code was reliably using the proxy without leaks. There were no good ways to test for this, so we made our own. SocksTrace (led by my colleague Robert Mindo) is an auditing tool that intercepts every network-related function call that an application sends to the Linux operating system; if any network-related function call isn't going through the proxy, SocksTrace can reject the connection request, and also log details that can help an application developer track down the bug.

We quickly realized that instead of simply rejecting the connection, we could also make SocksTrace reroute the leaking traffic so that it did go over Tor. This provided an easy way to make non-Tor-compatible applications run over Tor.

When we showed a prototype of SocksTrace to a Tor developer (including a demo of SocksTrace trivially surfacing a privacy bug in Firefox that the Tor developers had needed to manually discover by a code audit 2 years earlier), they were sufficiently impressed that they opened a GitLab ticket on the Tor issue tracker, proposing that Tor Project look into automatically running SocksTrace on Tor's web browser every time the browser code was updated. Namecoin has funding from NLnet Foundation [8] to move SocksTrace closer to such a deployment.

6 Project Case Study 3: Occlumask and Preventing Doxing

While proxy leaks are one way that Tor users could accidentally be deanonymized, another is harder to prevent: what the user talks about while anonymous. For example, if the user is in a chat room and mentions that it started snowing yesterday, that simple fact can narrow down the user's location. A few slip-ups like this can be all that's needed to figure out who they are. This isn't just a theoretical concern: a whistleblower was arrested in 2012 after he casually mentioned some personal anecdotes (e.g. political protests he had attended), and this narrowed down the set of suspects enough that it became economically feasible to catch him with targeted surveillance [10].

The Whonix Project, which specializes in high-quality documentation on how to stay anonymous, has a detailed list of things that you shouldn't discuss while anonymous. Unfortunately, humans are not psychologically very good at vigilantly avoiding a topic that they find interesting. We wanted to produce a better way to protect these users, without needing them to never slip up.

Occlumask (led by my colleague Alice Margatroid) is a tool that could be embedded in chat applications or web browsers, which uses an LLM ("large language model" – similar conceptually to ChatGPT, but running locally rather than the cloud for privacy reasons) to evaluate the topics being discussed in a conversation, and warns the user if they type something that might dox them.

While LLM's are routinely used for use cases where they make little to no sense (not unlike blockchains), Occlumask is a good example of a use case that plays to the strengths of an LLM. LLM's are reasonably good at reasoning about the topic of a conversation, and the risk of occasional hallucinations isn't a big problem – Occlumask tells the user what text is identified as risky, and if the user determines that Occlumask is mistaken, the user can simply ignore it.

Occlumask is still in quite early stages (our funding for Occlumask was only granted by Power Up Privacy [9] a few months ago), but it has already shown an impressive ability to notice deanonymization potential in text that we had missed before Occlumask pointed it out.

7 Correspondence to the Botball Skill Set

Many skills I picked up in Botball have been crucial to the privacy/security research I do at Namecoin. My previous two GCER papers [1][2] cover some of these, including:

- 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.
- Working within the Botball parts list is excellent practice for developing tools that need to run in resource-constrained environments.
- Writing Botball documentation and GCER papers is excellent practice for writing blogposts and conference talks that keep users and funders informed about what you're working on.
- Botball students who fork KIPR's open-source code and contribute their improvements are gaining excellent practice for open-source Git workflows.
- Botball's friendly nature (with teams encouraged to befriend nominal competitors) is excellent practice for real-world collaboration with other open-source projects that could be considered your competitors.

You'll probably find those two papers interesting if you're enjoying this one so far and would like to learn more about those parallels. Below are some additional parallel skills I've observed that weren't covered in those two papers.

7.1 Testing Limits of Your Parts

In Botball, the hardware has certain tolerances that you have to deal with. For example, trying to run a motor at a ticks-per-second velocity that's too high will impair the accuracy of your movement, while trying to use a rangefinder at distances that are too close or too far will impair your ability to sense objects. Many Botballers, before committing to a specific usage of such a hardware part, will do some empirical testing to see at what point the part ceases to work as intended. (Norman Advanced did this for the "ET" infrared rangefinder, and my understanding is that DeWitt Perry did something similar for motor velocity.)

We encounter the same issue at Namecoin from time to time. As a recent example, Encaya's subversion of the serial number field carries a hidden risk: is there a limit on the length of the serial number? For currently deployed cryptography (based on elliptic curves), signatures are relatively short: typically less than 100 bytes, which should fit without problems. Unfortunately, elliptic curve cryptography is

nearing the end of its usefulness: quantum computers [11] (once they exist) will be able to trivially solve the math problems whose difficulty is critical in making elliptic curve cryptography secure. The leading candidate for a signature scheme that is quantum-resistant is the amusingly-named CRYSTALS-Dilithium cryptosystem [12]. The problem here is that CRYSTALS-Dilithium signatures can potentially be in the vicinity of 10 kilobytes. Will TLS implementations allow such a long serial number? Legitimate serial numbers (that aren't carrying disguised data) certainly won't be that long, so it wouldn't be surprising if some software rejects them. The relevant specifications don't specify any limit, leaving it up to the implementation.

One of my most recent activities at Namecoin consisted of writing a tool to generate thousands of TLS connections, with the serial number length incrementing by 1 byte each time, to see at what point different software starts to complain. This work is ongoing; we hope to have answers soon.

7.2 Working Effectively Under High-Stress Emergencies

Botballers know this situation all too well: you're at the tournament, practice is over in an hour or two, and your bots are failing to score points. Perhaps it's due to a subtle difference between your practice board and what KIPR has set up. Norman Advanced and Team SNARC have had such conundrums countless times. We got pretty good at making changes to our bots with minimal test runs. A major component of this was being able to focus when under a time limit and dealing with unexpected failure modes, without getting too stressed or making stupid mistakes in our hurry.

Namecoin has run into this kind of situation as well. In one case, when the Heartbleed emergency [13] was publicly disclosed, we had to quickly gauge whether we had any users who were affected. Namecoin had some code that was vulnerable to Heartbleed, but it was not enabled by default, and we didn't know how many users might have enabled it. Gauging this was critical in determining whether we needed to rush out a patch immediately, or take our time doing more thorough testing of a fix. My colleague Ryan Castellucci solved this for us: they wrote up a tool that spidered the Namecoin network, looking for any users who had the affected feature enabled and publicly exposed. Within an hour, Ryan had determined that they could find exactly zero affected users at the time of the scan. This gave us some much-needed leeway to get a fix properly tested and deployed.

In another case, my colleague Cassini noticed that Namecoin mining seemed to be much slower than typical that day, and wondered if something had broken for a bunch of miners. Mining problems could be a source of security problems if left unfixed. The next day, a miner filed a bug, reporting that his node was reporting strange error messages. We were able to use the error logs he provided to deduce that someone had submitted a nonstandard transaction to the Namecoin network, which triggered an edge case that caused miners to get stuck. We were able to work with several miners to coordinate a manual workaround that got the nonstandard transaction mined at high priority, thus un-gumming-up the miners' operations. The "mining brownout" thus was fixed quite quickly, before any security problems manifested.

In a third case, I was contacted by a vulnerability disclosure coordinator about a non-public Bitcoin vulnerability that they believed also affected Namecoin; we had a few days to fix it before the vulnerability was scheduled to be publicly disclosed. We were able to apply Bitcoin's fix easily, but the

bigger problem was that Tor Project was using some of our code, and they had their own build and release schedule. My colleague Yanmaani and I therefore had to spend several hours carefully auditing Tor Project's usage of our code. We both independently assessed that while the vulnerable code itself was present in Tor Project's releases, all of the code paths that could trigger the vulnerable code were unreachable – when I had worked with Tor Project on that code, I had removed a lot of unneeded code to save space on the download, and it turned out that the code I had removed included all of the code paths that could trigger the vulnerability. We stayed in contact with Tor Project during our assessment, and everyone was relieved when we reported to the Tor developers that no urgent action was necessary on their end.

Being able to focus on the task without getting too stressed was a critical component of why were able to get these resolved smoothly without incident.

7.3 Cross-Pollination of Ideas

It's pretty common for Botballers to see another team's bots do something interesting, and as a result, come up with an idea for another application of that idea. This has shown up for me in Namecoin too.

My "disguise a signature as a serial number" trick was useful for onion services because the serial number is sent as part of the TLS protocol handshake, meaning we didn't have to retrieve the signature from some external source. It wasn't immediately obvious where else we could retrieve such a signature. I hadn't previously needed this trick when working with Namecoin's core activity (domain names on the Namecoin blockchain), because here it was easy: the TLS public key could just be stored in the blockchain next to the IP address.

However, once I had come up with this trick for onion services, I realized that this trick was still useful for the blockchain use case: it would allow us to save some storage in the blockchain by moving the TLS public key out of the blockchain. Blockchains are very storage-constrained (since all the historical data of every domain name needs to stick around forever), so this was a quite attractive benefit. I wouldn't have thought of this without having collaborated with Tor Project on TLS for onion services.

7.4 Fundraising and Grantwriting

Most Botballers are familiar with fundraising. Paying for a Botball kit costs money, as does traveling to GCER. Botball teams tend to learn certain skills for asking for money.

Namecoin resembles a nonprofit in funding structure: we don't get revenue from our users as a condition of using our software; our funding is largely from donations and grants. I tend to be Namecoin's lead grantwriter, and a major reason I'm comfortable doing it is that I had practice fundraising in Botball. Being able to explain to grantmaking organizations what it is that we're doing, in terms that they understand and find interesting and meaningful, takes practice. Botballers are well-placed to do well here.

7.5 Mentoring

It's not rare for former Botballers to wind up mentoring Botball teams. By all accounts, I seem to have been a pretty popular Botball mentor; the students, teachers, and parents at Alcott Middle School thought I was excellent at explaining the necessary skills that Botball students needed, without giving them all the answers (teaching through questions is a highly useful method). In parallel, when I interned at KIPR, the KIPR staff were excellent at mentoring me. KIPR encouraged me to not simply blindly follow instructions, and instead suggested that I become their resident expert on the specific topics that my internships specialized in. This went very well, and allowed me to build up both expertise and confidence that I wouldn't have gotten just from following instructions.

At Namecoin, I've mentored two new developers, both of whom are still working with us today. I modeled my mentoring approach for Namecoin interns after how I mentored Botballers, and how KIPR staff handled my internships. This was highly successful. I tended to teach through questions; the developers I mentored got to become our resident experts on specific topics, and built confidence as a result. One intern I mentored was under the Tor Project internship umbrella; he was one of four Tor interns that season (the only one being mentored by Namecoin), and he ended up being the only one of those four interns to successfully complete his project and present it to the full Tor Project staff.

I wouldn't have been able to mentor successfully if not for my prior experience in Botball.

7.6 Handing Off Project Ideas

Some Botballers have an interesting problem: they have more ideas for projects than they could possibly find time to work on all of them. This is a good problem to have, and Botballers who are fortunate enough to be plagued by this problem frequently solve it by handing off project ideas to teammates who either have a better skill set for a given project, or have more time to work on it, or both.

This was me at Norman Advanced, and it's still me in Namecoin today. Both SocksTrace and Occlumask originated as brainstorms I came up with, but I didn't have the time or skill set to do them justice. So I handed them off to Robert and Alice, so that the projects could be handled properly. I still have some involvement with SocksTrace and Occlumask, but I'm not the project lead on either, which is good for everyone involved.

7.7 Extracurricular Projects

While most students join a Botball team because they want to work on competition bots, it's common for Botball teams to find extracurricular projects to work on. For example, Norman Advanced has partnered with university researchers on a climatology research side project that has nothing to do with competition [14].

This also happens in privacy/security research. Namecoin's nominal focus is putting domain names on a blockchain, but as I describe above, we've worked on lots of side projects. In addition to Encaya, SocksTrace, and Occlumask, we've also worked on a virtual smartcard library (which another developer later reused for running smartcards over the Internet) [15], created a sandboxing tool for

restricting which CA's are valid for which websites [16], and contributed code to a TLS test suite maintained by Netflix [17].

Sometimes people get confused why Namecoin does all of these side projects, but I think it'll be self-evident to Botballers: Botball isn't just there for the competition, and Namecoin isn't just there for the blockchain.

8 Conclusion

Botball's well-known technical skills (e.g. C programming and embedded systems) are certainly useful in the real world, but many under-appreciated skills (such as the ones described in this paper) turn out to be highly important as well. I'm pleased that GCER sometimes hosts papers and presentations analyzing the various aspects in which Botball influences its former students in the real world, both in robotics and in the rest of STEM. Sadly, these benefits are still probably not very well-known outside of Botball circles, so I encourage readers to share these kinds of success stories about former Botballers with anyone who values STEM but may not understand Botball's educational strengths.

And, of course, I would be remiss without a career note to Botballers: if you think privacy should be a human right, and you think any of the topics I've discussed here are interesting, we'd love to have you at Namecoin. The excessive hype attached to blockchain-related projects has mostly died down in the last few years (mostly migrated to AI), but the actually-interesting projects in the space are very much still around and active. Open-source development is a great resume builder, especially open-source work done prior to graduating high school. Feel free to email me if you're potentially interested (`jeremy@namecoin.org` or `jeremyrand@danwin1210.de` – send to both addresses). It's OK if you don't view yourself as an expert in security topics – I certainly didn't think I was when that special awards judge suggested it to me in 2009, but, well, here I am.

9 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] Jeremy Rand. 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). GCER 2022. https://ynqrnzvuotdgvqj2g73f5tyj4rcvwutdemu23jgadawihbkc3lxkbwid.onion/pdf/gcer/2022/Namecoin_GCER_2022.pdf

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

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

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

[6] Jeremy Rand. Tor Meeting 2024 / GPN 22 / MoneroKon 4 Summary. https://www.namecoin.org/2024/07/24/tor-2024-gpn-22-monerokon-4-summary.html

[7] Jeremy Rand. 38C3 Summary. https://www.namecoin.org/2025/03/30/38c3-summary.html

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

[9] Power Up Privacy. https://powerupprivacy.com/

[10] U.S. Department of Justice. United States of America v. Jeremy Hammond. https://www.justice.gov/archive/usao/nys/pressreleases/March12/hackers/hammondjeremycomplaint.pdf

[11] Wikipedia Contributors. Quantum Computing. https://en.wikipedia.org/wiki/Quantum_computing

[12] CRYSTALS Team. Dilithium. https://pq-crystals.org/dilithium/index.shtml

[13] Codenomicon. Heartbleed Bug. https://heartbleed.com/

[14] Pranav Jayachand, Chris Albert, Reza Torbati, et al. Underwater-ROV-Project. https://github.com/normanadvanced/Underwater-ROV-Project

[15] The Namecoin Project. pkcs11mod. https://github.com/namecoin/pkcs11mod

[16] The Namecoin Project. certinject. https://github.com/namecoin/certinject

[17] Ian Haken. BetterTLS. https://bettertls.com/