# Namecoin as a Decentralized Alternative to Certificate Authorities for TLS: The Next Generation

Jeremy Rand
Lead Application Engineer, The Namecoin Project
https://www.namecoin.org/

OpenPGP: 5174 0B7C 732D 572A 3140 4010 6605 55E1 F8F7 BF85

Presented at 35C3 Monero Assembly

# A brief introduction to Namecoin

- Like the DNS, but secured by a blockchain.
- Uses the “.bit” top-level domain.
- Names are represented by special coins.
- First project forked from Bitcoin (in 2011; Bitcoin was created in 2009).
- Original focus of developers was on censorship-resistance.
  - We later became interested in PKI use cases (e.g. for TLS) as well.

# Getting rid of Certificate Authorities (CA’s) in TLS

- TLS trusts over 1000 certificate authorities.
  - CA’s get compromised.
    - DigiNotar (allegedly by Iranian intelligence).
  - We want to replace CA's with Namecoin.
- Subject of this talk: improvements in Namecoin integration with TLS implementations over the last year.
  - Better compatibility and scalability.
  - Smaller attack surface.

# Positive VS Negative Overrides

- We have 2 orthogonal goals:
  - Positive overrides: a certificate that matches the blockchain should be accepted, even if it's not signed by a public CA.
  - Negative overrides: a certificate that **doesn't** match the blockchain should be rejected, even if it **is** signed by a public CA.

# The State of Namecoin TLS One Year Ago: Positive Overrides

- “Dehydrated certificates”: deterministically reconstruct a self-signed end-entity X.509 certificate from only:
  - The domain name.
  - The public key.
  - The validity period.
  - The signature.
- The resulting certificate can be safely imported as a trust anchor.
- Certificate is 255 bytes of JSON; easily fits in Namecoin.

# The State of Namecoin TLS One Year Ago: Negative Overrides

- Abuse HPKP by setting a key pin for all .bit domains, which no one has the private key for.
  - We used a public key hash of 1/pi, scaled to 256 bits.
- This causes all public-CA-issued certs for .bit domains to be rejected.
- But our dehydrated certs will still work, because user-defined trust anchors are exempt from HPKP.

# Positive Overrides: Adding Firefox Compatibility

- Firefox uses the mozilla::pkix certificate validation library, with NSS as the trust store.
- mozilla::pkix doesn't honor end-entity trust anchors from NSS's trust store.
  - The Mozilla people believe that supporting this would be a footgun.
  - This means that dehydrated self-signed end-entity certs, though valid in NSS, aren't valid in mozilla::pkix.
  - How can we fix this?

# Name Constraints

- Name constraints restrict the set of domain names that a TLS CA can issue certs for.
- Intended use cases include:
  - A corporate intranet CA can be constrained to only issue certs within a corporate intranet TLD.
  - You can buy a name-constrained CA from a public CA, and then you can issue as many certs as you want within your domain name without bothering the public CA.

# Name constraints aren't often used

- Because:
  - Corporate intranet CA's like to violate employees' privacy.
  - Public CA's would rather make you buy more certs from them.
- Most high-profile user is probably Let's Encrypt: the Let's Encrypt CA has a name constraint preventing it from issuing certs for US military domain names.
- Most major TLS implementations do support name constraints.
  - Last straggler was probably Apple products, which only added support in the last few years.
  - See the BetterTLS test suite from Netflix for more details.

# Storing name-constrained CA's in the blockchain

- We can construct a name-constrained root CA trust anchor from a public key + domain name.
- Validity period... who cares, root CA's keep their keys offline, they don't need to rotate keys like TLS servers do.
- Signature... could be a self-signature, except that the name constraints RFC says that name constraints are ignored for self-signed CA's.
  - No idea if implementations follow the spec on this, but easy to workaround by signing with a locally generated root CA.
  - Avoiding a self-signature also saves blockchain space.

# Name-constrained CA's work fine in Firefox

- But what if we also want to support the (rare) TLS implementations that don't support name constraints?
  - Reuse the public key from the name-constrained CA to construct an end-entity cert, signed by the CA.
  - Load validity period and signature from blockchain just like Gen-1 dehydrated certs.
  - Load the resulting end-entity cert as a trust anchor for implementations that don't support name constraints.
  - Total blockchain storage identical to Gen-1 dehydrated certs.

# Negative Overrides without HPKP

- HPKP is on its last legs.
  - Chromium already deprecated it.
  - Firefox has an open bug for deprecating it.
  - RFC authors have abandoned it.
- So we need a new way to do negative overrides.

# Name constraints for negative overrides

- Let's say that we want to prevent all public CA's from issuing .bit certs.
  - We could politely ask them to put a name constraint in their cert, like Let's Encrypt did for .mil.
  - But they'd probably say no.

# Name constraints for negative overrides

- Can we force TLS implementations to pretend that those CA's have a name constraint?
  - Kind of, not really.
- There's a spec called "attached extensions" for this purpose.
  - Only implemented in a few GNU/Linux distros e.g. Fedora.
  - Via the p11-kit project.
  - No real-world software knows how to read this data from p11-kit.
  - Not implemented in CryptoAPI, NSS, OpenSSL, or basically anything else.
  - Also deletes any pre-existing name constraints in those CA's... might be bad news for .mil.

# Can we edit the trust anchor's certificate ourselves?

- We'd break the self-signature.
- We'd probably have to merge the Namecoin name constraint with any existing ones.
  - I'm not dumb enough to try to code that.
- Not clear whether name constraints even have an effect for self-signed certificates (such as most trust anchors).

# Can we edit the trust anchor's certificate ourselves? (2)

- What if we just replace the issuer and signature of the root CA, so that it's signed by a CA we locally created that has the name constraint we want?
- This is actually something that CA's do for each other all the time, it's called cross-signing.

# Cross-signing tooling issues

- All the existing tooling (e.g. openssl command line) requires that you have a Certificate Signing Request, **signed by the CA you want to cross-sign**.
- This is security theater that's not backed by any actual cryptography.
  - There's no cryptographic procedure by which I can prevent you from signing my public key.
  - The OpenSSL devs who added that requirement should feel bad.

# So I made my own tooling

- I made a Go library and command-line tool to cross-sign an input CA with a locally generated CA that has a name constraint.
  - Without the input CA's permission.
- Deployment:
  - Untrust the existing CA.
  - Trust the new locally generated CA.
  - Insert the new cross-signed CA to the intermediate CA store.

# Other use cases for my tooling

- You don't trust your corporate intranet CA to not MITM your Internet traffic?
  - You can use my tool to constrain that CA to only your intranet's domain names.
- Or you can apply a name constraint to all the public CA's so that they can't MITM your corporate intranet.

# Wrapper program for NSS: tlsrestrict_nss_tool

- I made a tool that wraps my name constraint tooling, and applies the name constraint to **all** built-in CA's in NSS.
- Run tlsrestrict_nss_tool, and you've got Namecoin negative overrides for Firefox (and the GNU/Linux version of Chromium).

# How does NSS's trust store work?

- NSS's trust store is split into 2 components:
  - CKBI
  - Softoken
- CKBI stores the immutable list of “built-in” trust anchors.
- Softoken stores all certificates inserted by the user.
  - To change the trust status of a CKBI certificate, you actually insert the CKBI certificate into Softoken, which will override the CKBI entry.

# Problems with using Softoken to store Namecoin TLS overrides

- Softoken uses BerkeleyDB or sqlite.
- BDB isn't concurrence-safe, so we can't edit trust settings while Firefox is running.
  - Makes it impossible to insert positive overrides based on hooking DNS queries.
  - Can (kind of) be worked around by pre-caching the entire set of TLSA records in the blockchain.
    - Only works with a full node.
    - Doesn't scale.
    - Doesn't work with DNSSEC delegation.

# Problems with using Softoken to store Namecoin TLS overrides (2)

- sqlite is slow.
  - tlsrestrict_nss_tool takes 8-9 minutes to apply name constraints to all of Fedora's trust anchors.
  - Partially because NSS is poorly optimized.
    - 2 sqlite transactions **per cert** that you touch.
    - I considered an LD_PRELOAD hook that intercepts sqlite commands and rewrites them to be more efficient.
    - Spent a couple of days designing such a hook.
    - Then came to my senses and realized that this approach is horrifying.

# Problems with using Softoken to store Namecoin TLS overrides (3)

- Race conditions when CKBI gets updated.
  - A removed CA might still have a constrained version in Softoken.
    - DNS domains are now vulnerable.
  - A newly added CA might not be constrained by Softoken at all.
    - Namecoin domains are now vulnerable.

# Problems with using Softoken to store Namecoin TLS overrides (4)

- Leaves private data (browsing history) in the NSS DB.
  - We try to delete old data after about an hour, but this isn't safe to rely on.
  - Softoken is not designed to be amnesic, we shouldn't be writing this data to the disk at all.
- Doesn't work at all in Tor Browser's default settings.
  - Because Tor Browser is competent, aims to be amnesic, and therefore completely disables Softoken and **only** uses CKBI.

# Problems with using Softoken to store Namecoin TLS overrides (5)

- Confuses key pinning.
  - Key pinning usually exempts user-defined trust anchors.
  - Works by storing a boolean “built-in” attribute for certificates.
  - Softoken doesn't allow setting the “built-in” attribute.
  - So if you replace all the CKBI certs with name-constrained versions in Softoken, the CKBI certs lose their “built-in” attribute, which disables key pinning enforcement.

# How exactly is NSS interacting with CKBI and Softoken?

- Turns out that CKBI and Softoken are actually just PKCS#11 modules.
- Yes, that's the spec that's usually used by HSM's.
- Could we just implement our own PKCS#11 module that handles Namecoin TLSA lookups without storing any state?
- We've done exactly that.

# Introducing ncp11: a PKCS#11 Module for Namecoin TLS

- Written in Go.
  - That means we had to write a Go library for creating PKCS#11 modules too.
  - Based loosely on Miek Gieben's PKCS#11 client library for Go.
- If you search ncp11 for a domain name, it looks up the trust anchor in Namecoin and returns it.
- It also proxies to CKBI, and edits any trust anchors returned by CKBI to add a name constraint blacklisting .bit.
- Kudos to aerth for developing a lot of the ncp11 code.

# Introducing ncp11: a PKCS#11 Module for Namecoin TLS (2)

- Works for any software that uses PKCS#11 trust stores.
  - i.e. anything based on mozilla::pkix, NSS, or GnuTLS.
  - Maybe other software too.
- The positive override functionality should be adaptable to work with non-Namecoin domains via DANE.
  - Negative override functionality is trickier, we're not sure if it's doable.

# ncp11 and Licensing

- ncp11 is a PKCS#11 module.
  - Which means it's a dynamically linked library (.so file).
- There is some disagreement about whether the GPL requires applications that use a GPL PKCS#11 module to be GPL-compliant.
  - FSF says it does.
  - The Namecoin developers think it probably doesn't (and that a copyright license can't do that even if it purports to).
  - Unfortunately there doesn't seem to be any case law on this.

# ncp11 and Licensing (2)

- We don't want to waste time arguing with FSF.
- We also don't want to waste time filing a lawsuit to get some case law.
- So, ncp11 is licensed under LGPLv3+, instead of Namecoin's typical GPLv3+.
  - Meaning you can use ncp11 with proprietary web browsers without violating the license.
  - But you should still use libre web browsers.

# ncp11 Workshop

- If you'd like to try out ncp11, I'm holding a workshop later.
- I'll cover setting up ncp11 for Chromium, Firefox, and Tor Browser.
- Check the Critical Decentralization Cluster schedule for details.
  - https://frab.riat.at/

# Sandboxing Namecoin TLS

- It really sucks that we need to give ncdns or ncp11 full access to the root CA trust store.
  - Means that if ncdns or ncp11 is compromised, it could enable non-.bit websites to be MITMed.
- Could we somehow sandbox ncdns or ncp11 to a "Namecoin-only root CA trust store"?
- We can actually do this with name constraints.

# Sandboxing Namecoin TLS (2)

- On ncdns/ncp11 install, generate and trust a CA with a name constraint whitelisting .bit.
- Give ncdns/ncp11 read access to the constrained CA's private key.
- Only give ncdns/ncp11 write access to the **intermediate** CA store.

# Sandboxing Namecoin TLS (3)

- For CryptoAPI:
  - Trust store is in the Windows registry.
  - Intermediate cert store is a different registry key from the Root cert store.
  - Registry has ACL's.
- For NSS:
  - NSS allows loading a PKCS#11 module while ignoring its trust flags.
  - This effectively makes the PKCS#11 module an intermediate-only cert store.

# Deficiency in HSTS

- HSTS (HTTPS Strict Transport Security) forces browsers to use HTTPS to connect to a given site.
- Enabled via an HTTP header.
- Can't protect you on your first visit to a website.
  - Because your browser hasn't seen that header yet.

# Introducing DNSSEC-HSTS

- WebExtension that redirects HTTP to HTTPS for websites that have a TLS certificate (DNSSEC-signed TLSA record) in their DNS.
- Works on first visit to website (unlike HSTS HTTP header).
- As decentralized as DNS.
  - No central repository of rules like HTTPS Everywhere.
  - Works with Namecoin.

# Contact Me At...

- https://www.namecoin.org/
- OpenPGP:
  5174 0B7C 732D 572A 3140 4010 6605 55E1 F8F7 BF85
- jeremyrand@airmail.cc
- Or just find me here at the Congress!  (The Namecoin logo on my shirt should help you find me.)