namecoin

Namecoin as a Decentralized Alternative to Certificate Authorities for TLS

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 34C3 Monero Assembly / Chaos West Stage

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).
  - CA’s achieve Too Big To Fail status.
    - Startcom, AKA the Martin Shkreli of Internet security.
- Subject of this talk: does Namecoin help us replace CA’s?

A brief survey of proposed solutions to the CA problem

- Trust agility (Convergence)
- A smaller set of trusted parties (DNSSEC/DANE)
- Limit window of opportunity for attackers (HPKP and CT)
- More accountability after a compromise (HPKP, CAA, and CT)

Were those really solutions?

- Note what's missing here: all of these “solutions” still allow a set of trusted parties to authorize MITM attacks.
- What we want is to be certain that a MITM attack will be detected **during the TLS handshake** without relying on a trusted 3rd party.
- Is this actually possible?

DNSSEC / DANE

- The DNS community long ago realized that a secure version of DNS could be used instead of CA’s.
  - Website owner puts a TLS certificate fingerprint in their DNS record.
  - End user’s browser makes sure that the certificate matches the fingerprint from DNS.
  - Standardized by IETF as DANE.
  - If we assume that DNS is secure (e.g. via DNSSEC), this should be secure.
- We don't trust the DNS, but maybe we do trust Namecoin to do what the DNS is supposed to do.

Adapting DANE to Namecoin

- Since Namecoin is interoperable with DNS, we can put TLS certificate fingerprints in Namecoin according to the DANE spec.
- A Namecoin-DNS bridge (running on localhost) signs the records with a bridge-generated DNSSEC key.
- User configures Unbound to use the bridge’s DNSSEC key for the .bit zone.
- Should be as simple as that, right?

Web browsers don’t support DANE

- No major web browsers do DNS lookups for DANE records.
- Some browsers intend to (eventually) support stapling of DANE records in the TLS handshake.
  - Useless for Namecoin, since for Namecoin the DNSSEC trust root is different per user.
  - Useless for preventing MITM’s, since this is only a positive override.
  - No ETA on browser support even for this.
- Chromium security team has flat-out refused to allow browser extensions to override cert verification results.

Overview of existing override methods that don't require browser vendor cooperation...

- Intercepting proxy, e.g. Convergence.
  - Hello SuperFish!
- Browser extension API's, e.g. DNSSEC-Validator.
  - Leaks cookies and more.
- Shared library hooks, e.g. CertShim.
  - Messes with unstable data structures (I don't trust this method to not corrupt memory).
- TL;DR all of these have problems.

Can we jerry-rig mainstream browsers to use Namecoin for TLS?

- Note there are 2 independent problems:
  - Positive override: a self-signed certificate needs to be accepted if it matches the Namecoin blockchain.
  - Negative override: a CA-signed certificate needs to be rejected if it doesn’t match the Namecoin blockchain.

TLS: Positive Overrides

- If you manually add a self-signed certificate to a browser’s trust store, it will be accepted.
- But this is a horrible idea for many reasons.
  - What if the certificate is also valid as a CA? Now it can impersonate other websites!
  - What if the certificate has multiple hostnames? Ditto!
  - Requires us to know the full certificate contents before we start the TLS handshake. TLS certificates are big – they won’t fit into a Namecoin record!

- <ryan-c> how small can we actually make a self-signed ecdsa cert?
- <Jeremy_Rand>Probably not small enough to fit in a Namecoin name
- <ryan-c> maybe not
- <ryan-c> er maybe it is
- <ryan-c> one sec
- <ryan-c> let me do some wizarding
- * Jeremy_Rand loves it when ryan-c puts on his wizard hat
- <ryan-c> Jeremy_Rand: the cert may too big, but we should consider cheating
- <ryan-c> Jeremy_Rand: yes, we can fit a self-signed ecdsa cert by cheating

Dehydrated Certificates

- Ryan's solution: starting with only a public key, validity period, signature, and hostname (called a **dehydrated certificate**), you can deterministically construct a valid certificate by filling a template (**rehydrating** the certificate).
  - Pubkey, validity period, and signature go in the Namecoin value.
  - Hostname determined by what Namecoin name is being looked up.
  - Use ECDSA instead of RSA – much smaller keys and signatures.

Efficiency Advantages of Dehydrated Certificates

- In theory: 104 bytes per certificate.
- In practice: 255 bytes.
  - Due to JSON/base64 encoding, no compressed pubkeys, other compromises.
- Before dehydration: 464 bytes binary, 620 bytes base64.
- A Namecoin name can hold 520 bytes (which also needs to include IP addresses and other DNS records).

Security Advantage of Dehydrated Certificates

- All of the potentially dangerous x509 fields (e.g. the CA bit) are controlled by the template, not the attacker.
- The only fields the attacker controls are the public key, the validity period, and the signature.
  - Attacker-controlled public keys are already standard in the TLS ecosystem – clearly safe.
  - Validity period's only potentially harmful effect is disincentivizing key rotation – only impacts the hostname who chose that validity period.
  - The signature check normally passes, and the only thing an attacker-controlled signature can change is making the signature check not pass – doesn't accomplish anything useful attack-wise.

Implementing Dehydrated Certificates

- I didn't want to use OpenSSL and friends.
  - API is impossible to use correctly.
  - I don't trust the memory safety of C/C++ code.
- Go has a nice x509 library.
  - API is simple.
  - Go is memory-safe.
  - Conveniently, Namecoin already was using Go for our DNS bridge implementation.

Implementing Dehydrated Certificates (2)

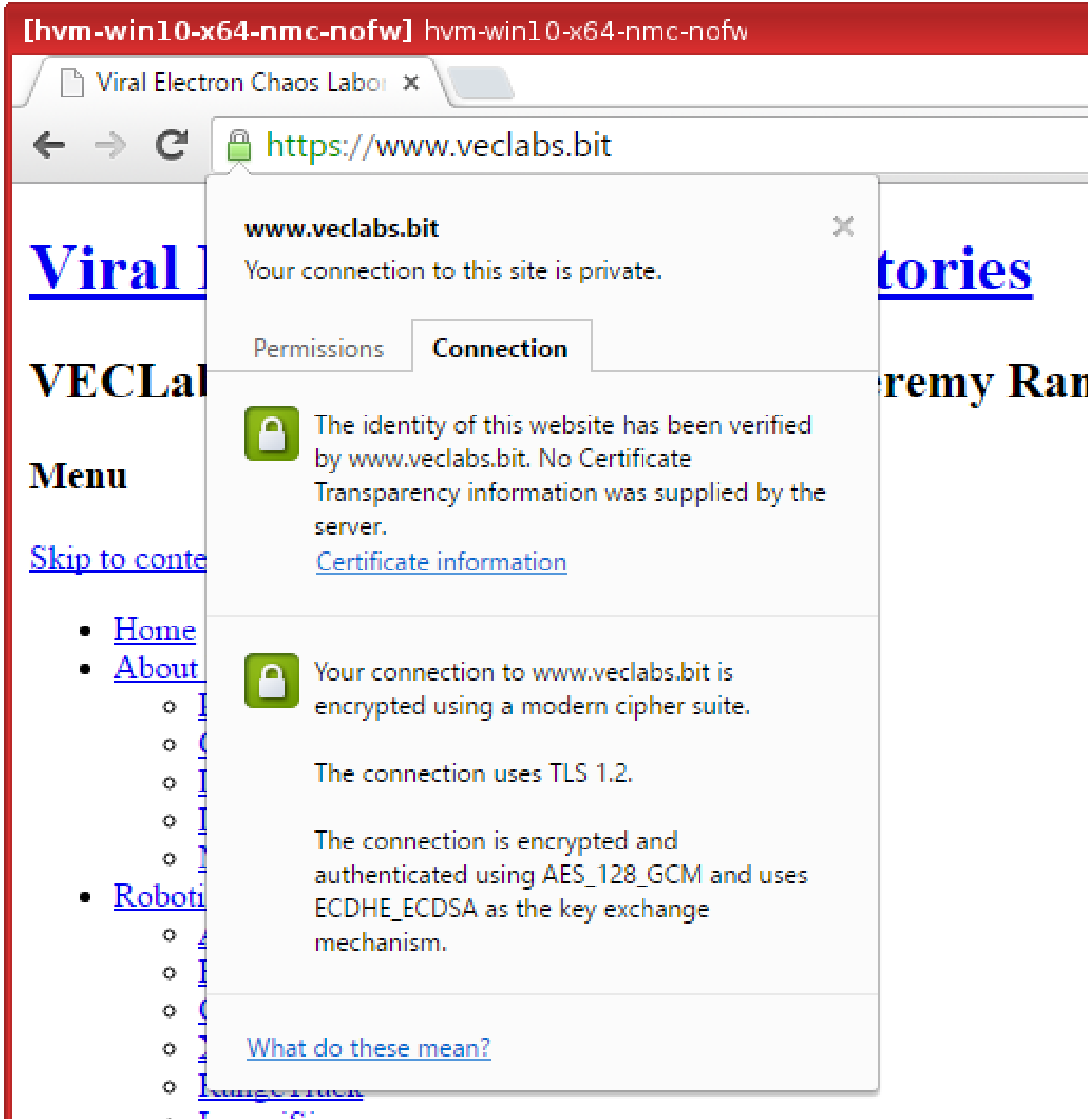
- Go's x509 API was actually a little bit *too* high-level – no publicly exported functions to splice a signature into a certificate.
- I ended up writing a “go generate” script that creates a copy of the standard library's x509 package, with an extra function added that uses private functions to splice the signature.
- Sadly, Go's standard library doesn't support compressed public keys for these curves. (So we're not saving as much space as we could be.)

Hooking it together

- When a DNS request for a Namecoin domain name is received by the Namecoin-DNS bridge on localhost, the dehydrated certificate is rehydrated into DER format, and injected into the trust store.
  - On Windows, using CryptoAPI certutil.
  - On GNU/Linux, using NSS certutil.
- Takes effect immediately for CryptoAPI and sqlite-based NSS.

(Side note: CryptoAPI reverse-engineering)

- CryptoAPI's certutil is slow, and often requires Administrator privileges.
- Turns out that CryptoAPI internally stores its cert store as blobs in the Windows Registry.
  - … with a custom undocumented binary blob format, not standard DER.
  - … and the reason for this custom format existing is so that hash operation results can be cached.
  - RSA and ECDSA operations aren't cached by this format.
  - Yes, this is an absurd design.
- Anyway, I wrote some Go code that can create these blobs and add them to the Registry – now the code is fast and doesn't need Administrator privs.



www.veclabs.bit

Your connection to this site is private.

The identity of this website has been verified by www.veclabs.bit. No Certificate Transparency information was supplied by the server.

Your connection to www.veclabs.bit is encrypted using a modern cipher suite.

The connection uses TLS 1.2.

The connection is encrypted and authenticated using AES_128_GCM and uses ECDHE_ECDSA as the key exchange mechanism.

TLS: Negative Overrides

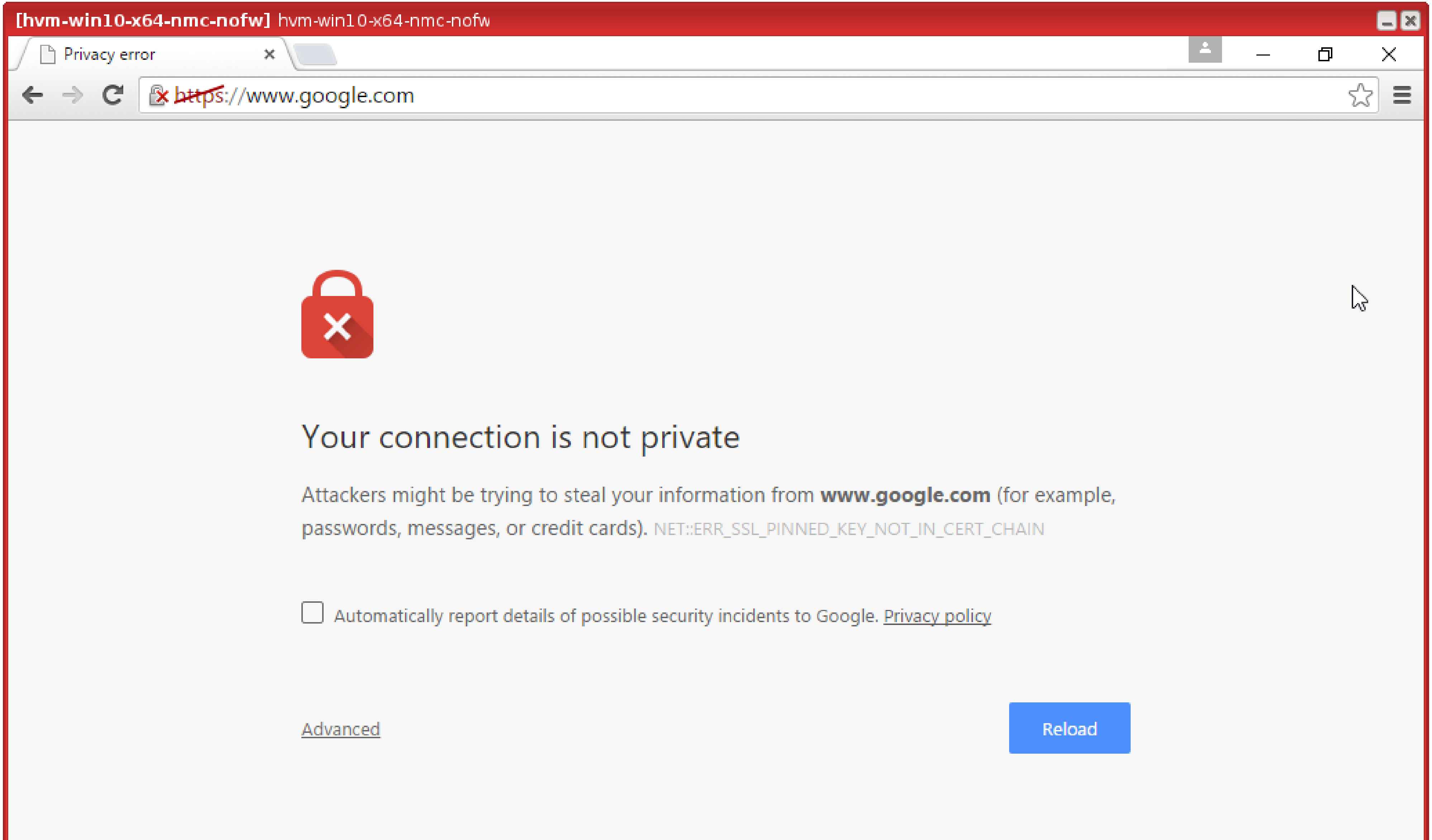
- So we got self-signed certs to be accepted if they match the blockchain... now we need to make sure that any CA-signed certs for Namecoin hostnames will be rejected if they don't match the blockchain.
- This turns out to be easier than the wizardry we needed for positive overrides.

Brief summary of HPKP

- HPKP tells a web browser to **only** accept certs for a given domain (possibly including subdomains) that match a whitelist of public key hashes.
- Hackers may want to intentionally MITM their own traffic without triggering HPKP errors – HPKP permits this by exempting user-defined CA's.
- Hmm… the self-signed certs that we added for positive overrides are considered user-defined CA's for the purpose of HPKP.

Abusing HPKP for our own ends

- What if we set the HPKP whitelist for “bit” (including all subdomains!) to a public key hash that no one has the private key for?
- All the user-defined positive override certs will still be valid.
- But all built-in CA's will no longer be trusted for Namecoin domains.
- Ryan suggested using 1/pi (scaled to 256 bits) as the nothing-up-my-sleeve public key hash.
- Turns out that Chromium stores its HPKP database as a JSON file in the profile directory; it's trivially easy to automatically add the needed entry when we install Namecoin.



[hvm-win10-x64-nmc-nofw] hvm-win10-x64-nmc-nofw
Privacy error
~~https~~://www.google.com
Advanced
Reload

Your connection is not private

Attackers might be trying to steal your information from **www.google.com** (for example, passwords, messages, or credit cards). NET::ERR_SSL_PINNED_KEY_NOT_IN_CERT_CHAIN

□ Automatically report details of possible security incidents to Google. Privacy policy

HPKP is disappearing from Chromium soon

- You might have heard Chromium is scrapping HPKP.
- For Windows, I *think* I can adapt the Windows key pinning features (e.g. EMET and Enterprise Certificate Pinning) to do negative overrides.
  - This has the benefit of working for all of CryptoAPI, not just Chromium.
- For GNU/Linux... no idea.

Mozilla Cert Override API

- It looks like Mozilla is tentatively willing to merge a cert override API to WebExtensions.
  - Subject to significant concerns about performance impact.
- I'm partway through coding a patch for this.
- Kudos to Mozilla for recognizing that this is an important use case.
  - Also thanks to the Mozilla people who've answered questions I've had while implementing that patch – especially David Keeler, Andy McKay, and Andrew Swan.

Currently Released Code

- Chromium/Windows support is working and released.
  - Go to https://www.namecoin.org , click “Downloads”, click “Beta Downloads”, download “ncdns Windows installer”.
- To test it, visit https://nf.bit (this is the Namecoin forum's Namecoin domain name).

Please help us end the insanity

- If you work on web browsers or other TLS implementations...
  - Please add API's for users to customize how TLS cert verification works.

Contact Me At...

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