TEU00311

What is the Internet doing to me? (witidtm)

Stephen Farrell
stephen.farrell@cs.tcd.ie

https://github.com/sftcd/witidtm
https://down.dsg.cs.tcd.ie/witidtm

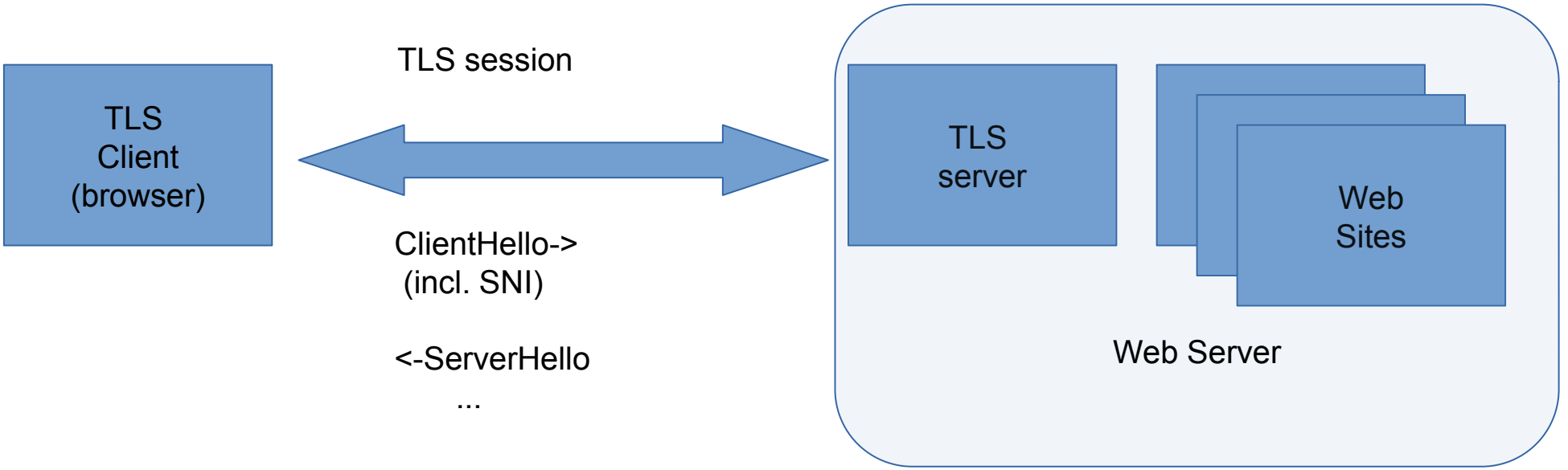
Overview

- As well as knowing about today's Internet it might be useful to know a bit about what's coming and how some things evolve
- So I'll describe some work I've been doing for the last couple of years, how that might go and how it might ultimately affect you
- That's a thing called "Encrypted ClientHello"

TLS and SNI

- Transport Layer Security (TLS, RFC8446) is the security protocol that secures the web and many other applications – HTTP running over TLS is what makes HTTPS
- One web server instance (e.g. an apache install using VirtualHost) can, and very frequently does, serve multiple web sites
- Each of those may (and is v. likely to) use different TLS server key pairs/certificates, which are the things that allow a TLS client (like your browser) to authenticate the web site, i.e. to know that you're really connecting to someone who (is related to someone who) controls "tcd.ie"

TLS for multiple web sites

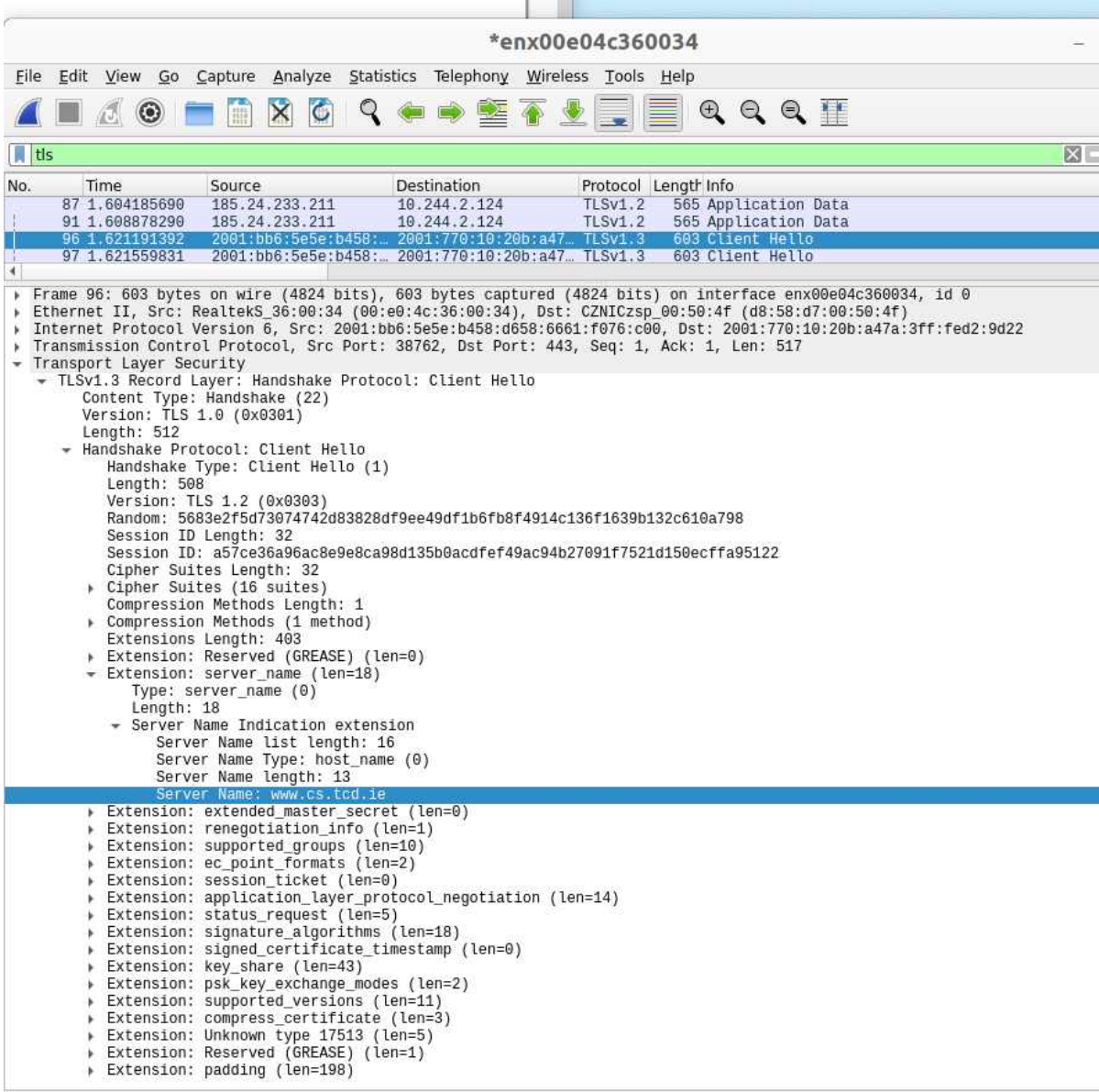


TLS and SNI

- One of the first things an HTTPS server needs to do is pick a TLS server key/certificate to use for the TLS session
- The client needs to verify that it's talking to the correct server via the TLS server certificate, which (for the web) contains the domain name of the web site
- Result: the first TLS message the client sends (the ClientHello) needs to specify the web site (DNS name) for which the TLS session is being established
- That's done using the Server Name Indication (SNI) extension to the ClientHello (RFC6066)

The SNI “Leak”

- Remember wireshark?
- It knows how to decode this kind of thing and we can se the cleartext SNI value on the right
- That ClientHello message was sent from a browser when I accessed https://www.cs.tcd.ie



SNI as a leak

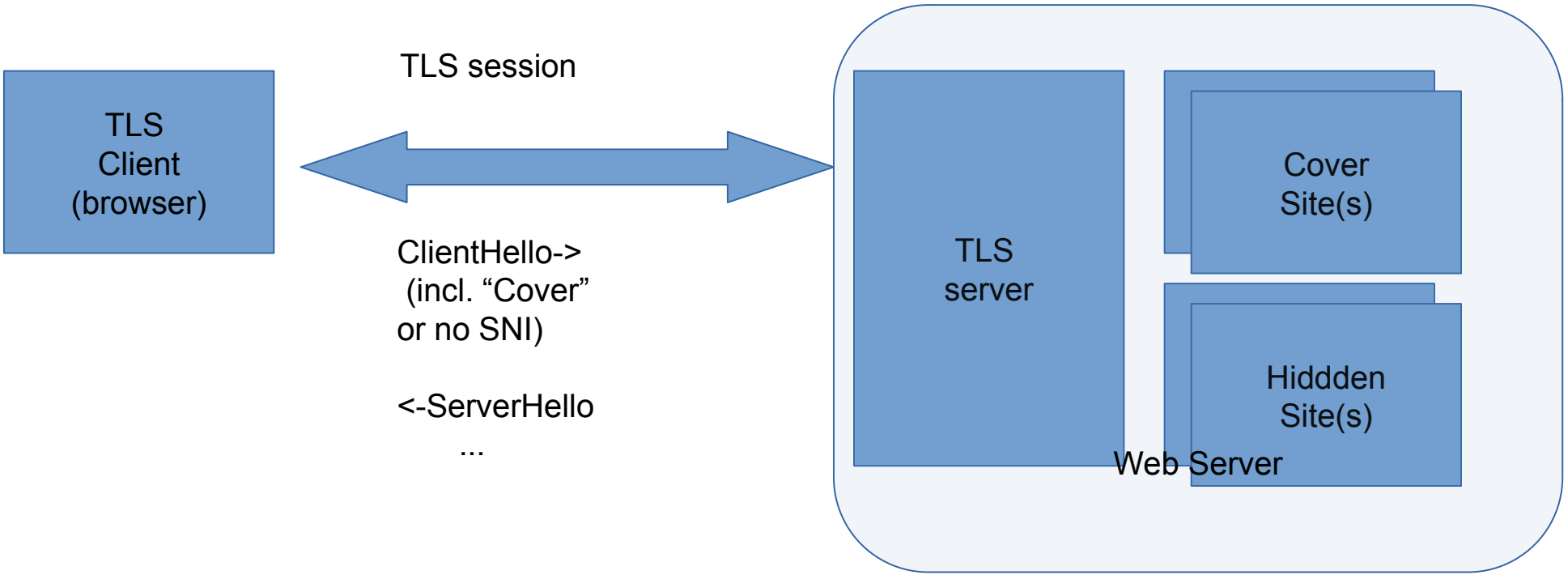
- Since the SNI value is sent in the first message, neither party has a key with which to encrypt, so SNI is sent in clear in the first TLS handshake message (the ClientHello)
- When accessing https://bank.example.com/getBalance the “getBalance” part will be encrypted (later, when the full HTTP request is sent) but the DNS name (“bank.example.com”) is sent in clear in the SNI extension
- That’s a noticeable leak, especially as the SNI is visible to everyone on the path (my ISP, the site’s ISP, every intermediate router, hosters, governments)
- SNI has also been used for censorship

  https://www.bleepingcomputer.com/news/security/south-korea-is-censoring-the-internet-by-snooping-on-sni-traffic/

SNI as a leak

- Domain “fronting” (where the SNI has the hoster’s name but the HTTP request has the “real” DNS name) was brittle and got turned off by service providers
- Previously, we weren’t motivated to try address this quite tricky problem because...
  - TLS server certificate was sent in the clear prior to TLS1.3
  - DNS name sent in clear using DNS protocol, but now we do have DNS privacy mechanisms (DoT/DoH)

What we'd like, and can do now ("co-located" variant)



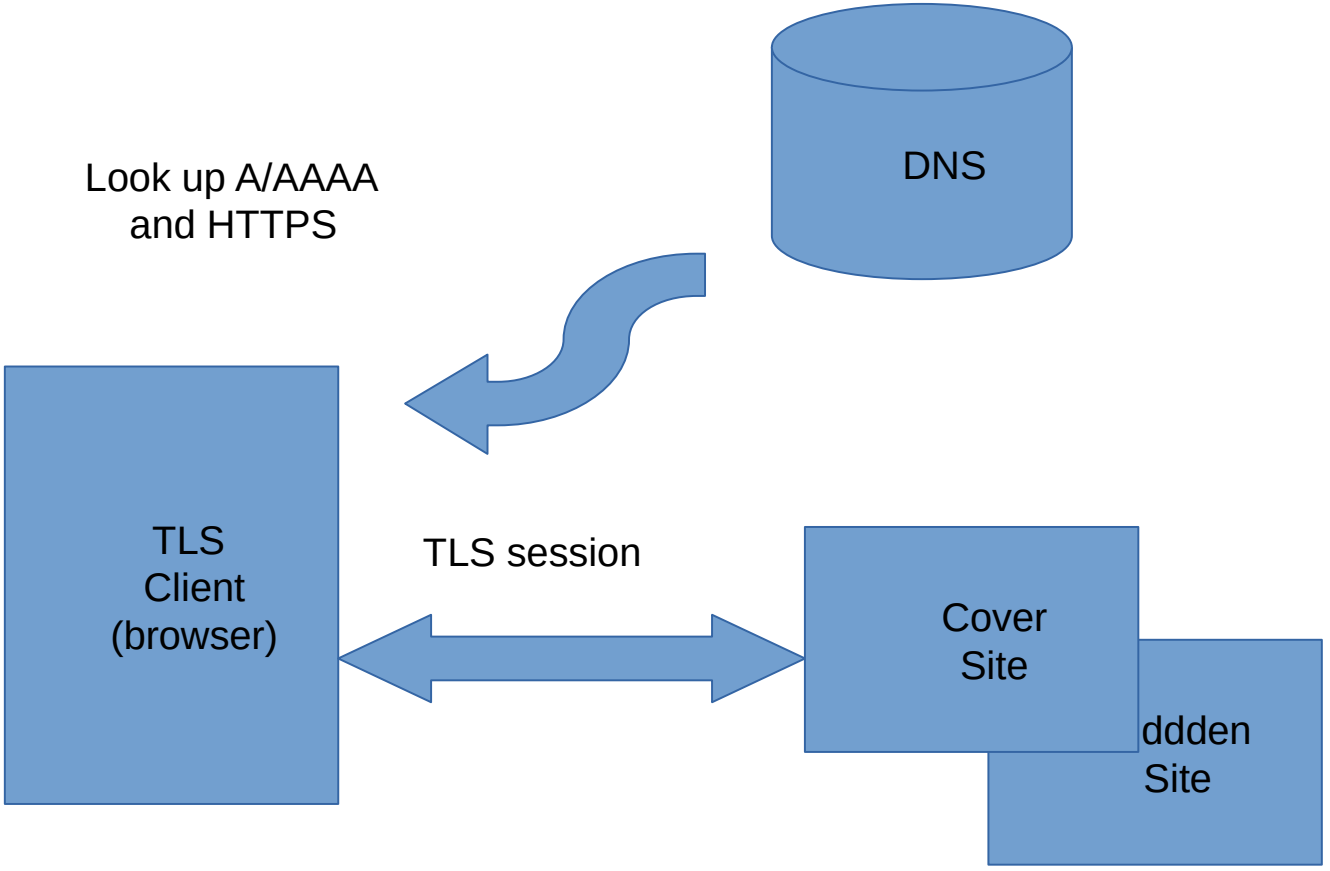
Encrypted ClientHello (ECH)

- Solution being developed in the IETF TLS WG: https://tools.ietf.org/html/draft-ietf-tls-esni
- Current draft is version -13, been in development since 2018
- Latest version seems comlpete, now at the stage of testing
- Multiple implementations exist, including mine
  - https://github.com/sftcd/openssl/ is my “fork” of OpenSSL - a very widely used TLS/encryption library that can be used with e.g. apache, nginx, lighttpd (web server implementations)

How does ECH work?

- Needs ability to create/consume new DNS resource records
- Needs TLS1.3 (earlier versions send server cert in clear)
- DNS privacy (DoT/DoH) not strictly needed but if you don’t use that maybe there’s less point in using ECH (Browsers will likely couple the two)
- Web site publishes a new public key/value in the DNS (“SVCB” or “HTTPS”) with some additional keys for ECH
- ECH-aware client (e.g. browser) can check if DNS record exists and has ECH keys
- All going well, use those ECH keys to derive a new shared-secret and send the “real” ClientHello message encrypted in side an “outer” ClientHello message
- The fact that ECH is being used is still visible

ECH Picture


DNS
Look up A/AAAA
and HTTPS
TLS
Client
(browser)
TLS session
Cover
Site
ddden
Site

GREASEing ECH

- The fact that ECH is being used is still visible
- That may be countered via “GREASEing” - having browsers that are not using ECH sometimes send a ClientHello that looks like it does use ECH
- GREASEing is an anti-ossification TLS implementation trick – clients and servers include garbage values for optional things in order to decrease the probability that middleboxes fixate on currently deployed protocol options - RFC8701

Interesting thing #1

- People who hate DNS privacy, hate ECH even more; for them, this is browsers and major web sites taking away information from which they've benefited
  - They used to monitor DNS queries but now DoT/DoH make that much harder
  - Switched to monitoring SNI from TLS ClientHello messages, and now ECH is killing that off
- They may have been using that information for what you consider good or bad (they considered it "good"):
  - Censorship, net-nanny, corporate policy enforcement, whitelisting TLS sessions to not MITM
- That DNS or SNI information could also be used for profiling or sold to advertisers, but I'm not aware of reliable information that that has happened

Interesting thing #2

- The people proposing/backing ECH are (I believe) doing so to try improve privacy (e.g. me, mozilla, aclu)
- ECH may however further increase centralisation since hiding in larger crowds is more effective
- It could be that the likes of cloudflare (test server deployed), apple (implemented client and server earlier) and google (implemented client so far) are the main beneficiaries of ECH – how should we consider that result?

Interesting thing #3

- ECH should make the kind of measurement Doug described harder, maybe a lot harder
- How do we try get accountability for software and systems without leaving open gaping holes to be exploited by attackers or nation-states?
  - Answer: we don’t know (or at least I don’t)

Interesting thing #4

- My guess is that whether or not, and how well, Google chrome implement ECH GREASEing will be the key determinant of whether or not ECH gets long term deployment
- As ECH sticks out, censors like the GFW can just block it
- That’s easy if it’s not widely used and would kill deployment
- If almost all ClientHello messages appear to contain ECH then such blocking gets harder
  - Maybe not for the GFW but perhaps for most other censors

Last interesting thing

- The processes for defining and implementing ECH are open to anyone with the skills and interest to get involved
  - Consumes a lot of time, but getting the skills required isn't really hard, all of you are certainly smart enough
- So you could (if you choose) end up involved in key decisions about how the Internet affects you that can reverberate up to fairly highly political levels
  - (Don't get a big head though:-)
  - That's part of the "permissionless innovation" I mentioned at the start
- It's also possible to be as involved with fewer technical skills, e.g. in civil society organisations or campaigning groups

Last slide for the module

- I hope you enjoyed the module, learned stuff and aren't too depressed about it all
- I'm really sorry that the technical community of which I've been part has produced this crap surveillance capitalism stuff
- But there are alternatives and practices you can adopt with not too much thought or work – do that! You are not at all near helpless!
- I also hope this annoys you sufficiently that you think about how to do something more about the bits you consider unacceptable

Thanks!

- Feel free to mail me anytime (next year, whenever) if there’s relevant stuff with which I can help