The Security Impact of HTTPS Interception

Zakir Durumeric, Zane Ma, Drew Springall,
Richard Barnes, Nick Sullivan, Elie Bursztein,
Michael Bailey, J. Alex Halderman, Vern Paxson

University of Michigan, University of Illinois Urbana-Champaign,
U.C. Berkeley, ICSI, Mozilla, Cloudflare, Google

Why is HTTPS Important?

Protects against network eavesdropping and man-in-the-middle attackers.

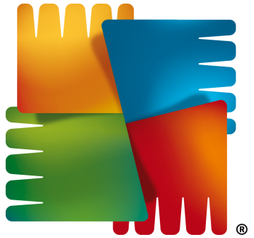
- Malicious access points / WiFi sniffing
- ISP traffic manipulation / ad injection
- Nation state attackers

HTTPS is on its way to becoming ubiquitous.

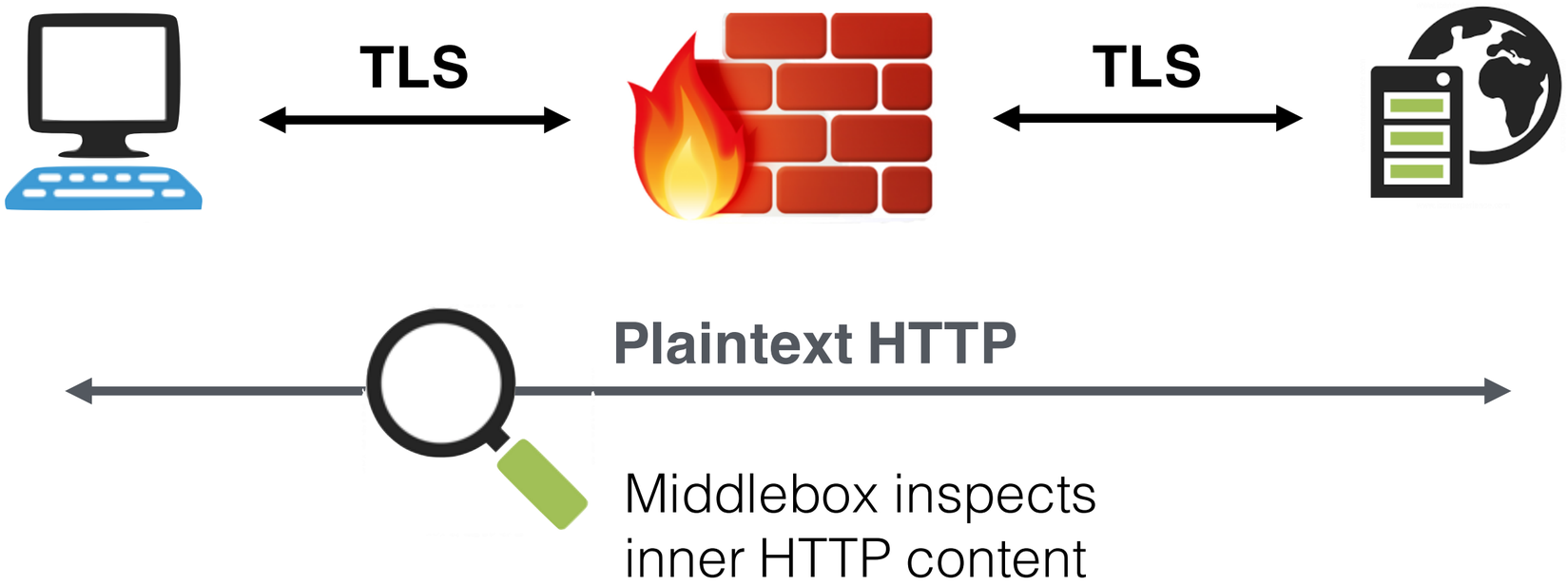
- 54M domains use Let’s Encrypt
- >60% browser connections use HTTPS
- Browsers eventually plan to warn on plain HTTP

HTTPS Interception

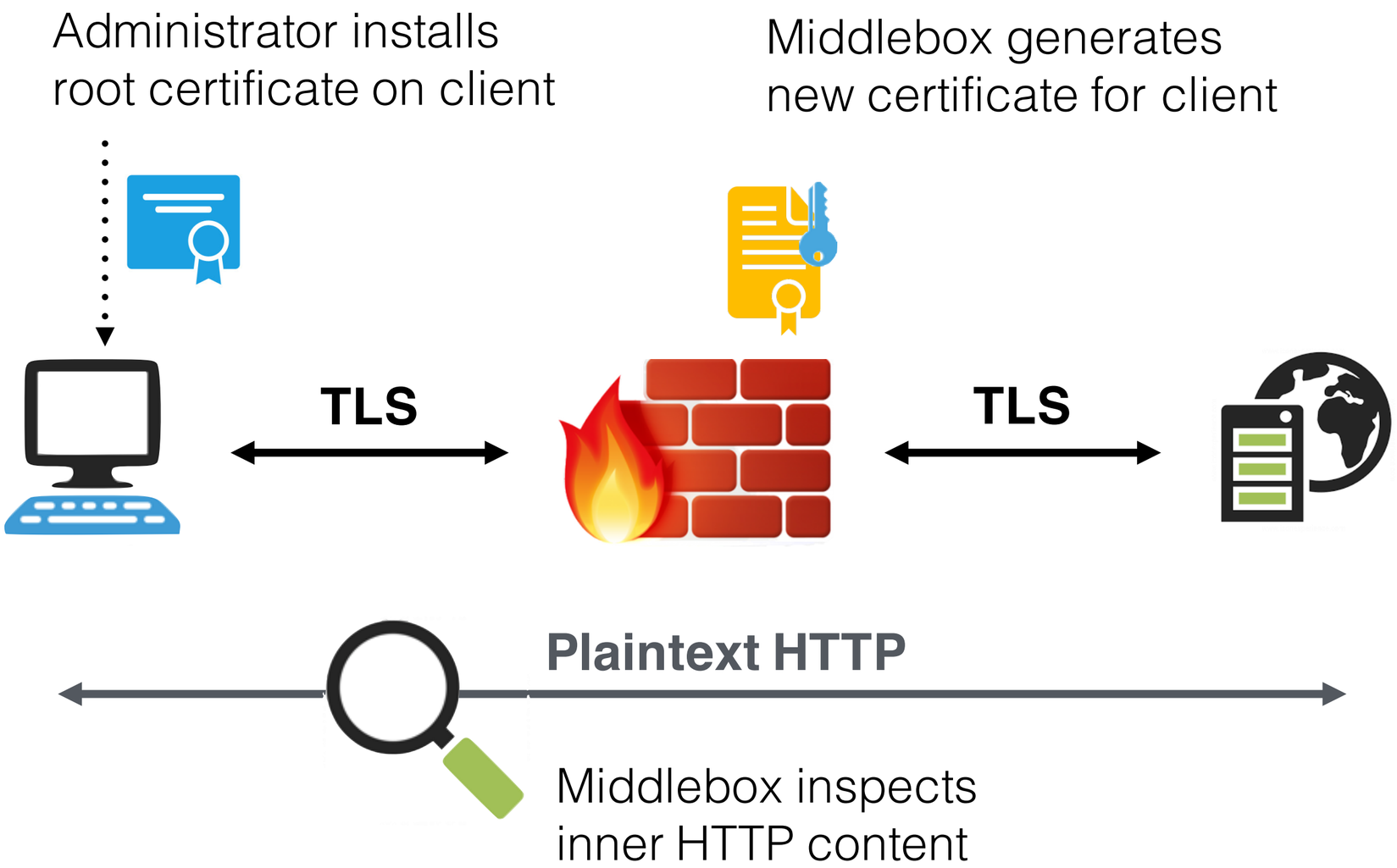
Middleboxes and security software are increasingly intercepting HTTPS connections in order to *inspect encrypted content*.



How HTTPS Interception Works



How HTTPS Interception Works



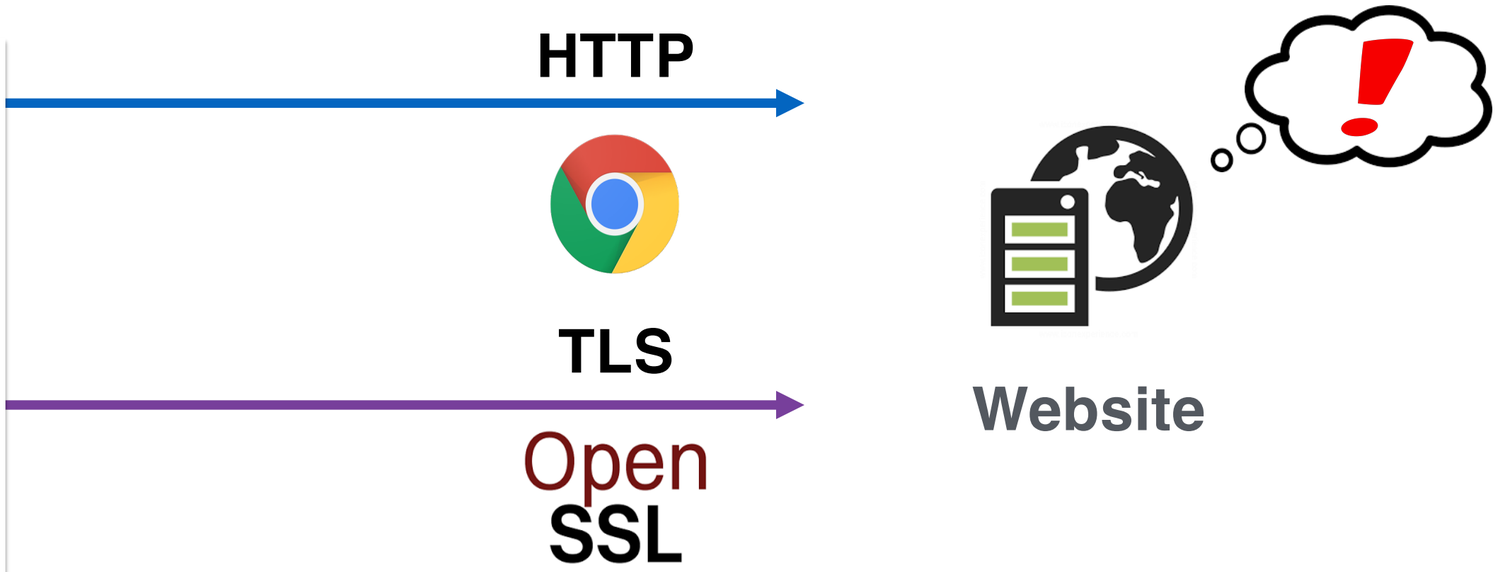
How do we measure the total amount of interception?

Change in TLS Library

Plaintext HTTP

HTTP User Agent: Chrome

Measuring Interception



Websites can potentially detect interception by identifying a *mismatch* between network layers

Fingerprinting Network Layers

HTTP

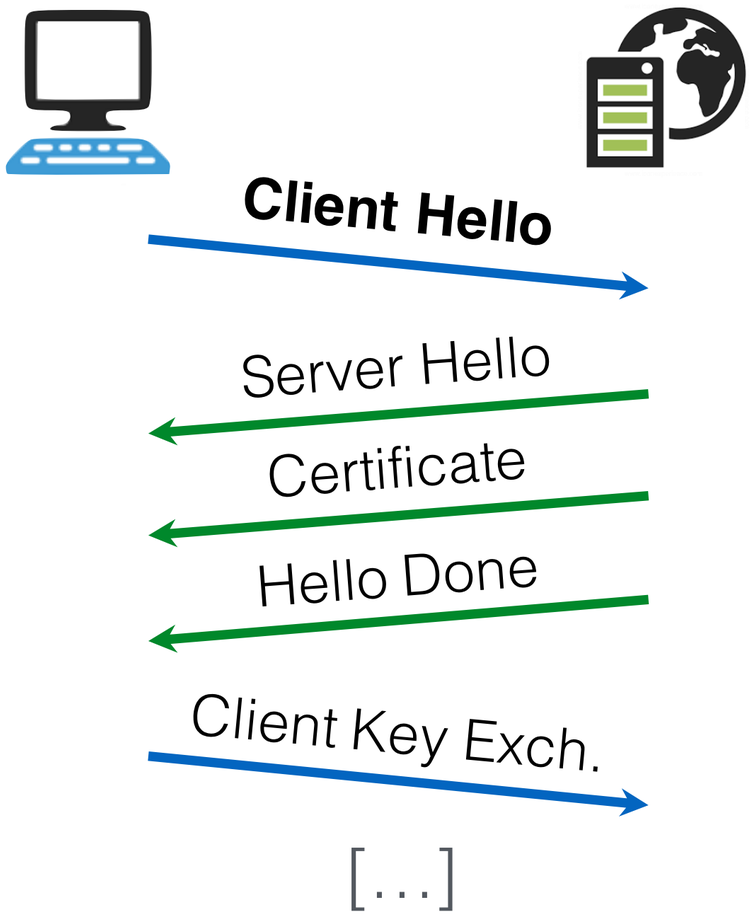
Parse HTTP User Agent Header:

```
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_2)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.95
Safari/537.36
```

TLS

No identifying field. Instead, we built a set heuristics that identify whether a TLS handshake is consistent with a browser.

Typical TLS Handshake




```
Secure Sockets Layer
▼ TLSv1.2 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 217
  ▼ Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 213
      Version: TLS 1.2 (0x0303)
    ▶ Random
      Session ID Length: 0
      Cipher Suites Length: 36
    ▶ Cipher Suites (18 suites)
      Compression Methods Length: 1
    ▶ Compression Methods (1 method)
      Extensions Length: 136
    ▶ Extension: Unknown 35466
    ▶ Extension: renegotiation_info
    ▶ Extension: server_name
    ▶ Extension: Extended Master Secret
    ▶ Extension: SessionTicket TLS
    ▶ Extension: signature_algorithms
    ▶ Extension: status_request
    ▶ Extension: signed_certificate_timestamp
    ▶ Extension: Application Layer Protocol Negotiation
    ▶ Extension: channel_id
    ▼ Extension: ec_point_formats
        Type: ec_point_formats (0x000b)
        Length: 2
        EC point formats Length: 1
      ▼ Elliptic curves point formats (1)
          EC point format: uncompressed (0)
    ▶ Extension: elliptic_curves
    ▶ Extension: Unknown 43690
```

Client Hello

Firefox vs. GnuTLS Client Hellos

Extensions
Server Name (SNI)
Extended Master Secret
Renegotiation Info
Elliptic Curves
[…]

Ciphers
ECDHE_ECDSA_AES128_GCM_SHA256
ECDHE_RSA_AES128_GCM_SHA256
ECDHE_RSA_CHACHA20_SHA2156
ECDHE_ECDSA_AES256_GCM_SHA384
[…]

Curves
secp256r1
secp384r1
secp521r1

Extensions
Extended Master Secret
Encrypt then MAC
OCSP Status Request
Server Name (SNI)
[…]

Ciphers
ECDHE_ECDSA_AES128_GCM_SHA256
ECDHE_ECDSA_AES128_GCM_SHA386
ECDSA_CAMELLIA_128_GCM_SHA256
ECDSA_CAMELLIA_128_GCM_SHA384
[…]

Curves
secp256r1
secp384r1
secp521r1
secp224r1
secp192r1

Investigating Common Products

We analyzed the TLS Client Hello messages from popular browsers browsers, middle boxes, client security software, and malware

Every product we investigated produced a unique TLS Client Hello message

Not always possible to identify product based on the handshake, but possible to detect whether a handshake is incompatible with a given browser

Deploying Heuristics

We deployed our heuristics for one week at three large service providers:

- Mozilla Firefox Update Servers
- Cloudflare CDN
- Popular E-commerce Site

Observed 7.75B HTTPS connections

Overall Interception Rates

We find a varying amount of interception between vantage points:

	No Interception	Likely Interception	Confirmed Interception
Cloudflare	88.6%	0.5%	10.9%
Firefox	96.0%	0.0%	4.0%
E-Commerce	92.9%	0.9%	6.2%

Overall Interception Rates

We estimate that 5-10% of all HTTPS connections are intercepted.

Firefox	96.0%	0.0%	4.0%
E-Commerce	92.9%	0.9%	6.2%

Measuring Security Impact

If interception products are performing high quality handshakes, there isn't an inherent security risk

We measured the security impact of interception by grading the security features advertised by the intercepted connection and the original browser

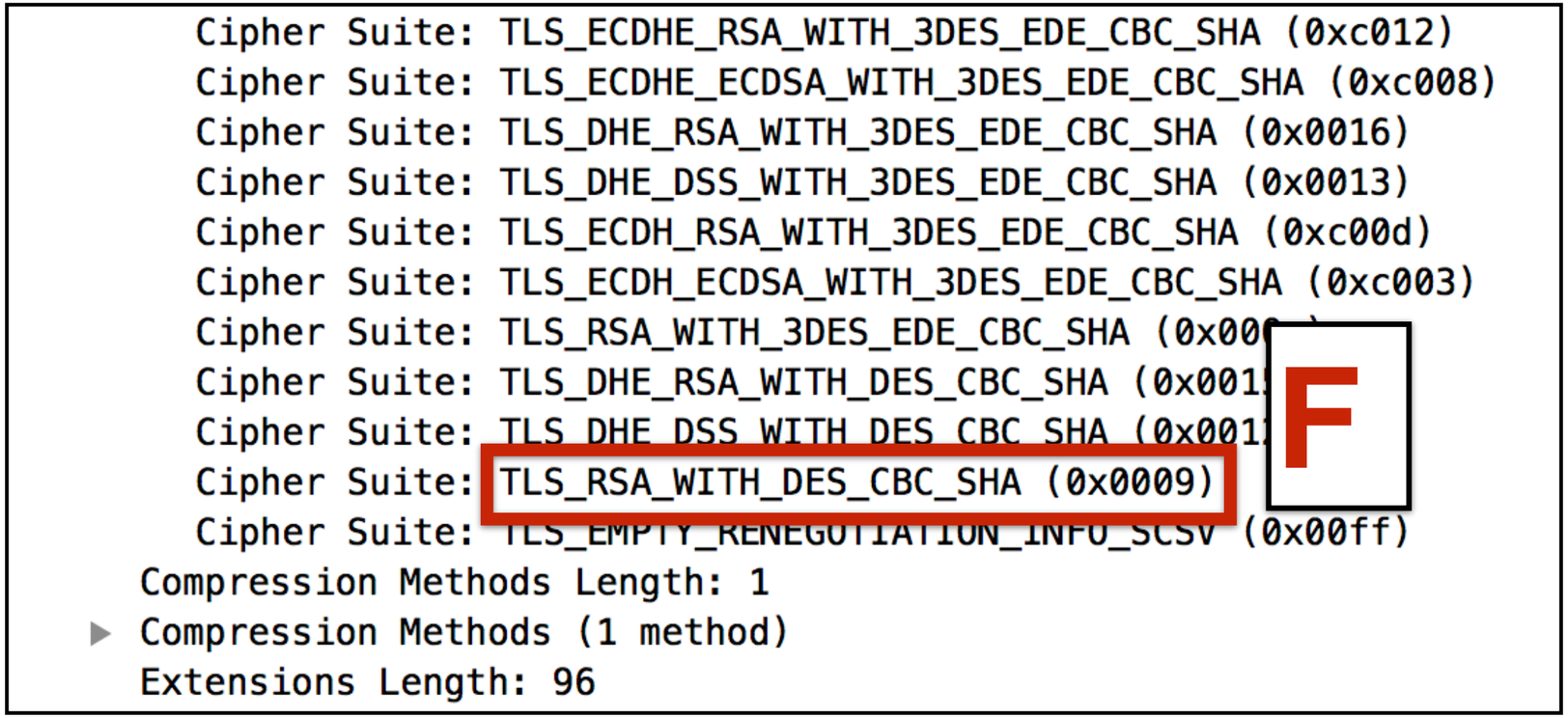
Quantifying Security Impact

We defined a security grading scale base on parameters advertised in Client Hello

Applied to original browsers and the connections we observed in the wild

Grading Scale	
A	**Optimal.** Equivalent to a modern web browser (e.g., Chrome)
B	**Suboptimal.** Non-ideal but not vulnerable to attacks
C	**Known Attack.** Vulnerable to known attack (e.g., RC4)
F	**Severely Broken.** An attacker could easily intercept connection

Security Grade Example



Security Impact of Interception

	Increased Security	Decreased Security	Severely Broken
E-Commerce	4%	27%	18%
Cloudflare	14%	45%	16%
Firefox Updates	0%	66%	37%

Security Impact of Interception

	Increased Security	Decreased Security	Severely Broken
E-Commerce	4%	27%	**18%**
Cloudflare	14%	45%	**16%**
Firefox Updates	0%	66%	**37%**

Middlebox Security

Network middleboxes have a worse security profile than client-side software

62% of connections are less secure

58% are severely broken

```
x-forwarded-for:
192.168.15.56

x-bluecoat-via:
abce6cd5a6733123
```

Why is security suffering?

Investigating Products

We investigated the default configurations of popular interception products:

- Popular middleboxes (e.g., A10, Bluecoat, Cisco)
- Antivirus software (e.g., Avast, AVG, Kaspersky)

We ran a series of automated tests against products

Security Profile of Interception Products

	Increased Security	Same Security	Decreased Security	Severely Broken
Client Security Products	0/20	2/20	18/20	10/20
Middleboxes	0/12	1/12	6/12	5/12

No products implemented new HTTPS features beyond the TLS specification (e.g., HPKP)

Defenses

Our fingerprinting library available on GitHub:
https://github.com/zakird/tlsfingerprints

Implemented in Caddy server, can warn users:

Caddy has the ability to detect certain Man-in-the-Middle (MITM) attacks on HTTPS connections that may otherwise be invisible to the browser and the end user. This means Caddy can determine whether it is "likely" or "unlikely" that a TLS proxy is actively intercepting an HTTPS connection.

Lots Blame to go Around

Security companies are acting negligently. Products designed to aid security add severe vulnerabilities.

Administrators need to test middleboxes to ensure that they are not downgrading security.

Client-side AV should never be intercepting HTTPS.
Can inspect content more safely within the browser.

Crypto libraries need secure defaults.
Currently difficult to lock down OpenSSL, etc.

Moving Forward

Security community needs to reach consensus on whether HTTPS interception is acceptable

If we're going to permit interception… we should investigate extending the TLS protocol to allow middleboxes to communicate with browsers safely

(e.g., mcTLS lets endpoints specify permitted middle boxes and authenticate each hop)

We should reconsider dependencies between HTTP and TLS that make secure interception products very hard to implement (e.g., HPKP)

Need to standardize certificate verification so that it can be implemented safely outside the browser.

Conclusion

We showed that web servers can detect interception by identifying a mismatch between network layers

We estimate that 5-10% of HTTPS connections are intercepted

As a class, interception products severely reduce the security of HTTPS connections

The Security Impact of HTTPS Interception

Zakir Durumeric, Zane Ma, Drew Springall,
Richard Barnes, Nick Sullivan, Elie Bursztein,
Michael Bailey, J. Alex Halderman, Vern Paxson

University of Michigan, University of Illinois Urbana-Champaign,
U.C. Berkeley, ICSI, Mozilla, Cloudflare, Google