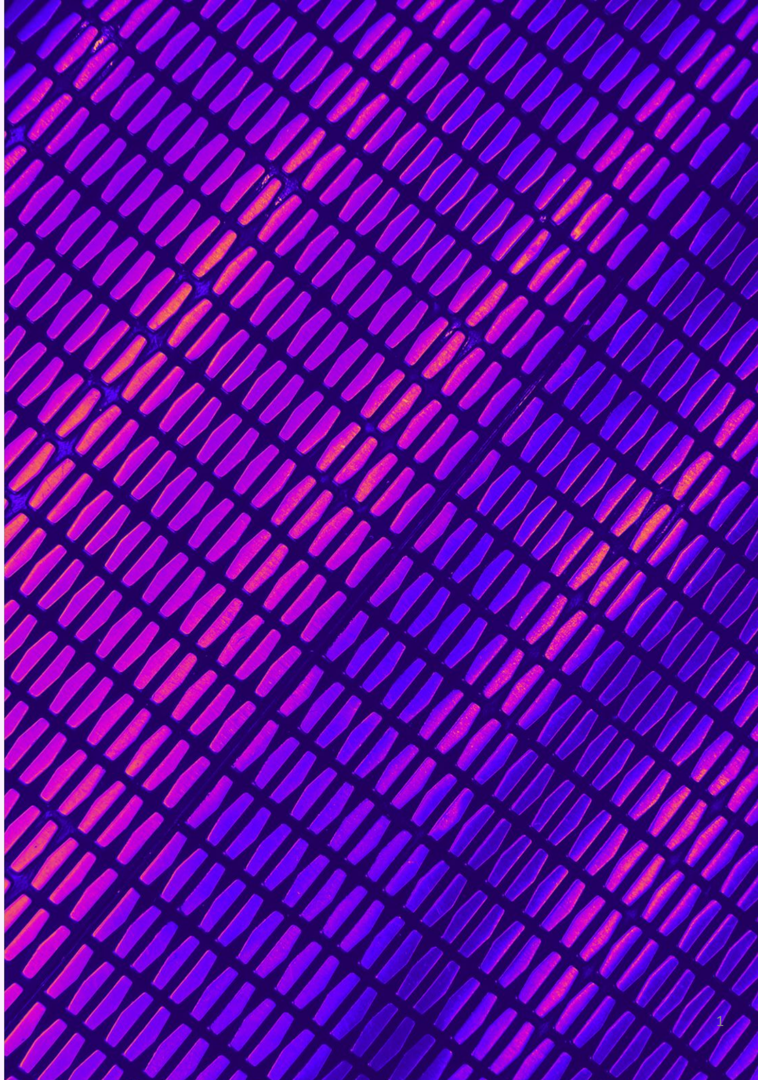
We *Really* Need to Talk About Session Tickets:

A Large-Scale Analysis of Cryptographic Dangers with TLS Session Tickets

Sven Hebrok, Simon Nachtigall, Marcel Maehren, Nurullah Erinola,
Robert Merget, Juraj Somorovsky, Jörg Schwenk

Presenter: Jacob Stolker



Key Takeaways

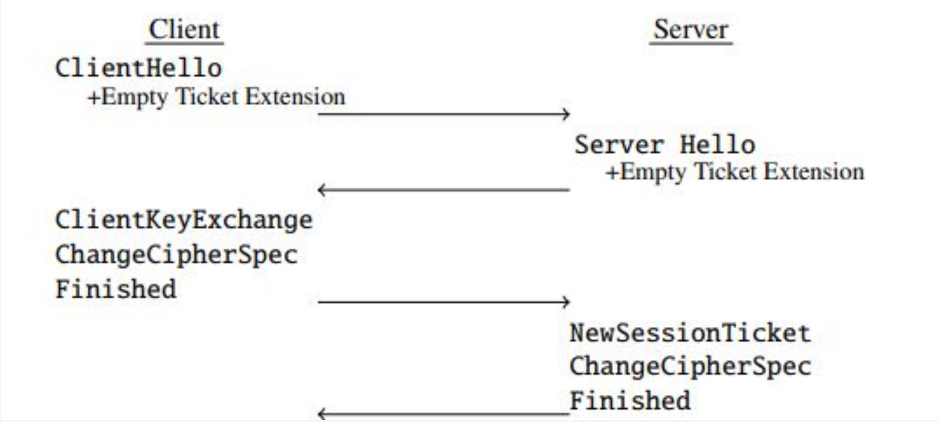
- TLS session tickets enhance performance, but can be vulnerable for several reasons
  - Lack of adherence to cryptography best practices
  - Poor maintenance and configuration of TLS servers

- Extensive scans can reveal vulnerabilities in TLS implementations

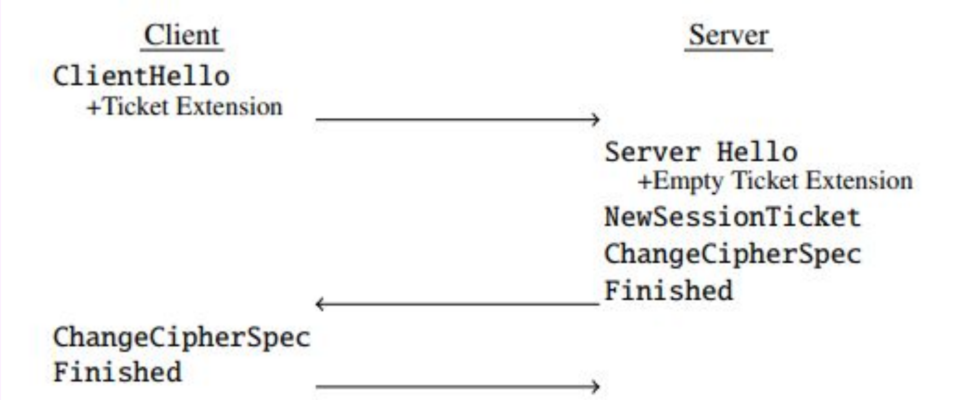
- A large number of AWS instances had some vulnerabilities with TLS security
  - ~1.9% of Tranco top 100k hosts had critical vulnerabilities

TLS Handshake

- Used to establish a client/server connection
- Resumption handshake allows faster reconnection



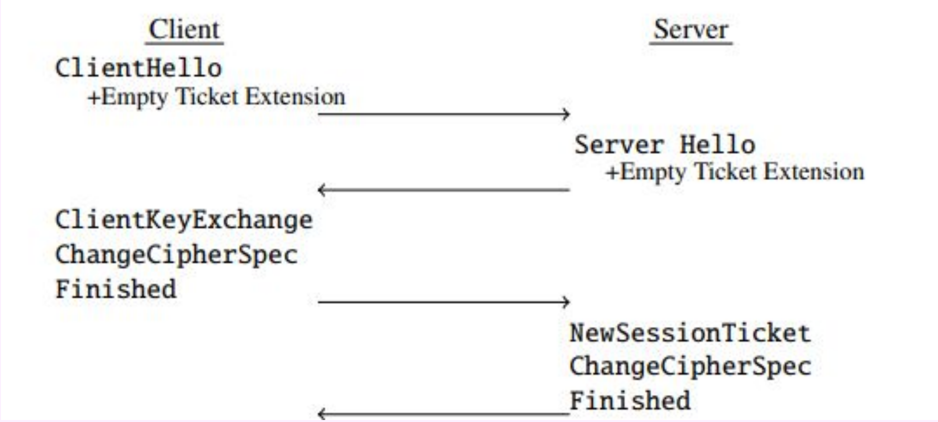
Standard TLS 1.2 handshake



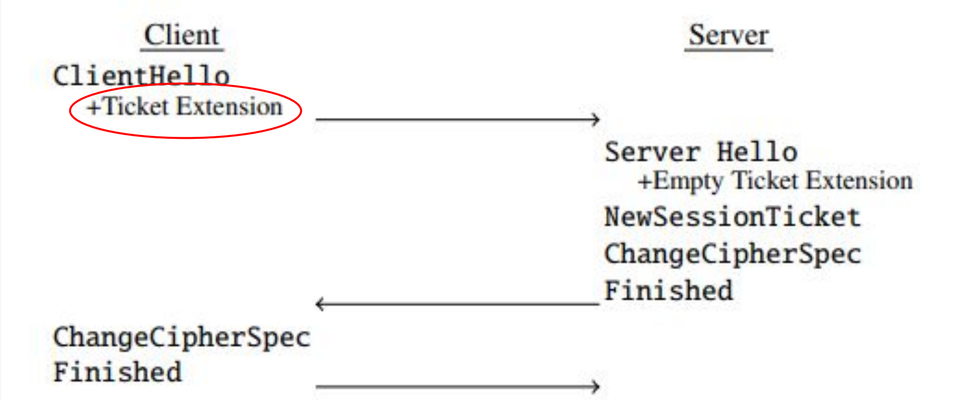
Standard TLS 1.2 resumption handshake

TLS Handshake

- Used to establish a client/server connection
- Resumption handshake allows faster reconnection



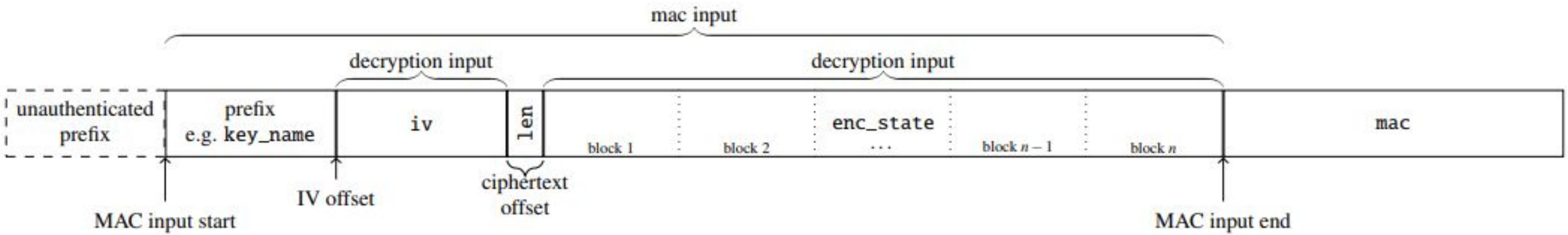
Standard TLS 1.2 handshake



Standard TLS 1.2 resumption handshake

TLS Session Tickets

- An encrypted and authenticated version of a TLS connection state
  - + other parameters
- Stored entirely by the client
  - no separate TLS server database required
- Allows resumption of connection
  - With half the time and 4% the normal cpu load



Session ticket format

Session Ticket Encryption Key (STEK)

- All session tickets are encrypted with STEK, which can be vulnerable
  - An attacker with the STEK can
    - Decrypt all session tickets (except with TLS 1.3 only future tickets)
    - Impersonate the server

Common Vulnerabilities

- Unencrypted session tickets
  - OpenPGP and S/MIME bugs
- Weak encryption keys
  - GnuTLS (all zero key)
- Reused keystream
  - Often occurs in counter-based cipher modes (GCM, CCM, CTR)
- Cryptographic wear-out
  - Probability of using the same nonce twice should be negligible
  - With AES-GCM, a 12B STEK should only be used 4.2 billion times
- Broken or weak authentication
- Weak or outdated algorithms
- Side channels (timing attacks)

Standardization

- RFC 5077
  - Recommended structure of session tickets

```
struct {
    opaque key_name[16];
    opaque iv[16];
    opaque encrypted_state<0..2^16-1>;
    opaque mac[32];
} ticket;
```

  - Recommended cryptographic standards
    - Encrypted with AES-128-CBC
    - Authenticated with HMAC-SHA-256

Analysis within Open-Source

		Session Ticket Format						Symmetric Algorithms	
Library	**Version**	`magic`[a]	`key_name`	`seed`[a]	`iv`[b]	`len`	`mac`	**Encryption**	**Authentication**
RFC 5077		–	**16**	–	**16**	**2**	**32**	**AES-128-CBC**	**HMAC-SHA256**
BoringSSL	2021[c]	–	**16**	–	**16**	–	**32**	**AES-128-CBC**	**HMAC-SHA256**
Botan	2.19.2	8	4	16	12	–	16	**AES**-256-GCM	(GMAC)
GnuTLS	3.7.6	–	**16**	–	**16**	**2**	20	**AES**-256-**CBC**	**HMAC**-SHA1
GoTls	go1.18.3	–	**16**	–	**16**	–	**32**	**AES-128**-CTR	**HMAC-SHA256**
MatrixSSL (TLS 1.2)	4.3.0	–	**16**	–	**16**	–	**32**	**AES**-256-**CBC**	**HMAC-SHA256**
MatrixSSL (TLS 1.3)	4.3.0	–	**16**	–	12	–	16	**AES**-256-GCM	(GMAC)
mbedTLS[d]	3.1.0	–	4	–	12	**2**	16	**AES-128**/256-GCM **AES-128**/256-CCM	(GMAC) (CBCMAC)
OpenSSL	3.0.3	–	**16**	–	**16**	–	**32**	**AES**-256-**CBC**	**HMAC-SHA256**
Rustls	0.20.6	–	–	–	12	–	16	ChaCha20	Poly1305
s2n	1.3.15	–	**16**	–	12	–	16	**AES**-256-GCM	(GMAC)
Apache	2.4.54	*Format of OpenSSL*						**AES-128-CBC**	**HMAC-SHA256**
Nginx	1.22.0	*Format of OpenSSL*						**AES-128**/256-**CBC**	**HMAC-SHA256**
OpenLiteSpeed	1.17.6	*Format of BoringSSL*						**AES-128-CBC**	**HMAC-SHA256**

a: These fields are only added by Botan.
b: IV or Nonce.
c: BoringSSL does not use releases. We analyzed the commit `dddb60e` from 2021-08-31.
d: mbedTLS can be configured to use different algorithms.

Analysis within Open-Source

		Session Ticket Format						Symmetric Algorithms	
Library	**Version**	magic[a]	key_name	seed[a]	iv[b]	len	mac	**Encryption**	**Authentication**
RFC 5077		–	**16**	–	**16**	**2**	**32**	**AES-128-CBC**	**HMAC-SHA256**
BoringSSL	2021[c]	–	**16**	–	**16**	–	**32**	**AES-128-CBC**	**HMAC-SHA256**
Botan	2.19.2	8	4	16	12	–	16	**AES-256-GCM**	(GMAC)
GnuTLS	3.7.6	–	**16**	–	**16**	**2**	20	**AES-256-CBC**	**HMAC-SHA1**
GoTls	go1.18.3	–	**16**	–	**16**	–	**32**	**AES-128-CTR**	**HMAC-SHA256**
MatrixSSL (TLS 1.2)	4.3.0	–	**16**	–	**16**	–	**32**	**AES-256-CBC**	**HMAC-SHA256**
MatrixSSL (TLS 1.3)	4.3.0	–	**16**	–	12	–	16	**AES-256-GCM**	(GMAC)
mbedTLS[d]	3.1.0	–	4	–	12	**2**	16	**AES-128/256-GCM** **AES-128/256-CCM**	(GMAC) (CBCMAC)
OpenSSL	3.0.3	–	**16**	–	**16**	–	**32**	**AES-256-CBC**	**HMAC-SHA256**
Rustls	0.20.6	–	–	–	12	–	16	ChaCha20	Poly1305
s2n	1.3.15	–	**16**	–	12	–	16	**AES-256-GCM**	(GMAC)
Apache	2.4.54	*Format of OpenSSL*						**AES-128-CBC**	**HMAC-SHA256**
Nginx	1.22.0	*Format of OpenSSL*						**AES-128/256-CBC**	**HMAC-SHA256**
OpenLiteSpeed	1.17.6	*Format of BoringSSL*						**AES-128-CBC**	**HMAC-SHA256**

a: These fields are only added by Botan.
b: IV or Nonce.
c: BoringSSL does not use releases. We analyzed the commit dddb60e from 2021-08-31.
d: mbedTLS can be configured to use different algorithms.

What was scanned?

- Pre-T1M
  - Preliminary tests of a portion of the T1M
- Tranco top 1M (T1M)
  - Regularly updated list of the top 1M most popular websites
- IP100k
  - Random 100k IPv4 hosts that responded on port 443 (https)
- IPF
  - Full IPv4 address range in August 2022

Scanning Methodology

- Online testing
  - Session tickets support
  - Authentication (accepts modified tickets)
  - Padding oracle attacks (try various block sizes)
- Offline testing
  - Common prefixes (prefix tree of a certain depth)
  - Unencrypted secrets (common bytes in multiple tickets)
  - Reused keystream (XOR two tickets)
  - Weak keys (brute force with a list)

Scanning Results

- Preliminary scans revealed a large number of AWS instances with weak STEK
- Vulnerabilities are rare, but easy to detect

Scan	Date	Tested Versions	Statistics			Offline Analysis			Online Analysis	
			Supports TLS	Issues Ticket	Resumes Ticket	Unencrypted Ticket	Weak STEK	Reused Keystream	Missing Auth. Protection	Padding Oracle
pre-T1M	2021-04	1.2	66,992	53,059	–	0	1,923	–	–	–
T1M	2021-05	1.2 – 1.3	760,293	594,238	547,159	0	3	–	–	–
T100k	2022-04	1.0 – 1.3	71,200	58,069	55,003	0	1	0	0	0
IP100k	2022-04	1.0 – 1.3	80,972	57,493	55,969	0	0	0	0	0
IPF	2022-08	≤1.2	39,390,365	29,621,531	–	0	189	1	–	–

Scanning results

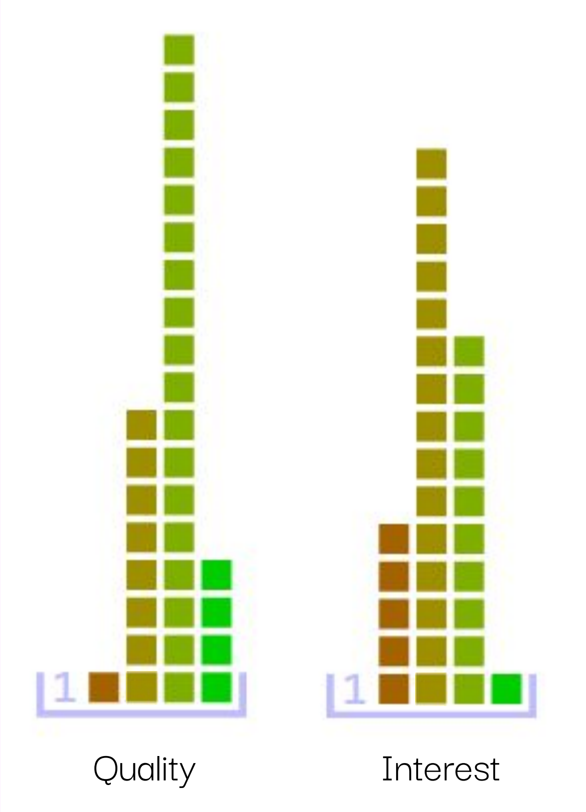
Related Works

- TLS Scanning
  - Public key exchange validation (Valenta et al.)
  - Looking for Bleichenbacher vulnerability (Böck et al.)
- TLS Key Entropy
  - Vulnerability of shared RSA primes (Heninger et al.)
  - Randomness low entropy in TLS (Hughes)
- Session Tickets
  - 10% of Alexa top million sites keep the same STEK for >30 days (Springall et al.)
  - 65% of all users can be tracked permanently by session tickets (Sy et al.)
  - TicketBleed: Extracting 31 bytes of uninitialized memory using tickets (Valsorda)

Discussion

- Does moving to TLS 1.3 help mitigate some of the vulnerabilities?
- Should there be an enforced standard for session tickets?
- Can MITM attacks be performed if a server's STEK is compromised?
- As a client, can we even know if session tickets are ill-formatted or poorly implemented?
- Should older, insecure algorithms continue to be allowed?
- How can we ensure keys are picked randomly and rotated consistently?
- Should we just do away with session tickets entirely? Is it not worth it the performance gains?
- Should we switch to session IDs?

Stats



Thank you!