DEEPSEC

TLS Forensik

Secure Linux Administration Conference 2019, Berlin

René Pfeiffer, DeepSec In-Depth Security Conference

Warum TLS?

- Sehr weit verbreitet
- Reichhaltiges Angebot von Bibliotheken
- „Leicht“ zu implementieren
- Hyper Transfer *Tunnelling* Protocol Secure („HTTPS”)
- TLS Traffic seit 2013 stark gestiegen
- Unterstützung von End-To-End-Encryption

Motivation

DEEPSEC

- Inspektion und Klassifikation von TLS Traffic
- Isolation von SSL Traffic (legacy systems)
- Klassifikation der Endpunkte
- Suche nach Anomalien / Mustern
- Sammeln von Indikatoren vor Tiefenanalyse
- Metadaten statt Big Data

Inspiration

DEEPSEC

- ThinThread
  - Programm der NSA zur Analyse abgefangener Daten
  - Identity Daten für Analyse unerheblich
  - Zuordnung zu Identitäten bei Bedarf
- Analogien
  - Inhalte der TLS Transmissionen unerheblich
  - Fokus auf Metadaten
  - Zugriff auf Inhalt möglich, wenn Client kontrolliert wird

Fingerprinting

DEEPSEC

- Fokus auf Metadaten / Protokolleigenschafen
- Erhebung sehr leicht
- nicht intrusiv / passiv möglich
- Freiheitsgrade
  - Nachricht-/Paketgröße in Bytes
  - Optionen (Art, Reihenfolge)
  - Zeitliche Abfolge (Zeitstempel)

Transport Layer Security

DEEPSEC

- Secure Sockets Layer (SSL) / Transport Layer Security (TLS)
- SSL veraltet, TLS v1.0/v1.1 ebenso
- TLS v1.2 hat Brückenfunktion
  - Kompromiss zwischen v1.0/v1.1 und Sicherheit
  - Perfect Forward Secrecy (Session Keys)
- TLS v1.3 seit August 2018 (RFC8446)
  - Grundlegende Überarbeitung
  - Neue Algorithmen
  - End-to-End Encryption per Design

TLS v1.3 Neuerungen

- ~~Session Renegotiation~~
- ~~Compression~~
- ~~CBC Mode~~
- ~~RC4, DES, 3DES, Export-Algorithmen~~
- ~~SHA1~~
- ~~MD5~~
- ~~DSA~~
- ~~Custom DH Groups~~
- ~~Statischer RSA Handshake~~

TLS v1.3 Neuerungen

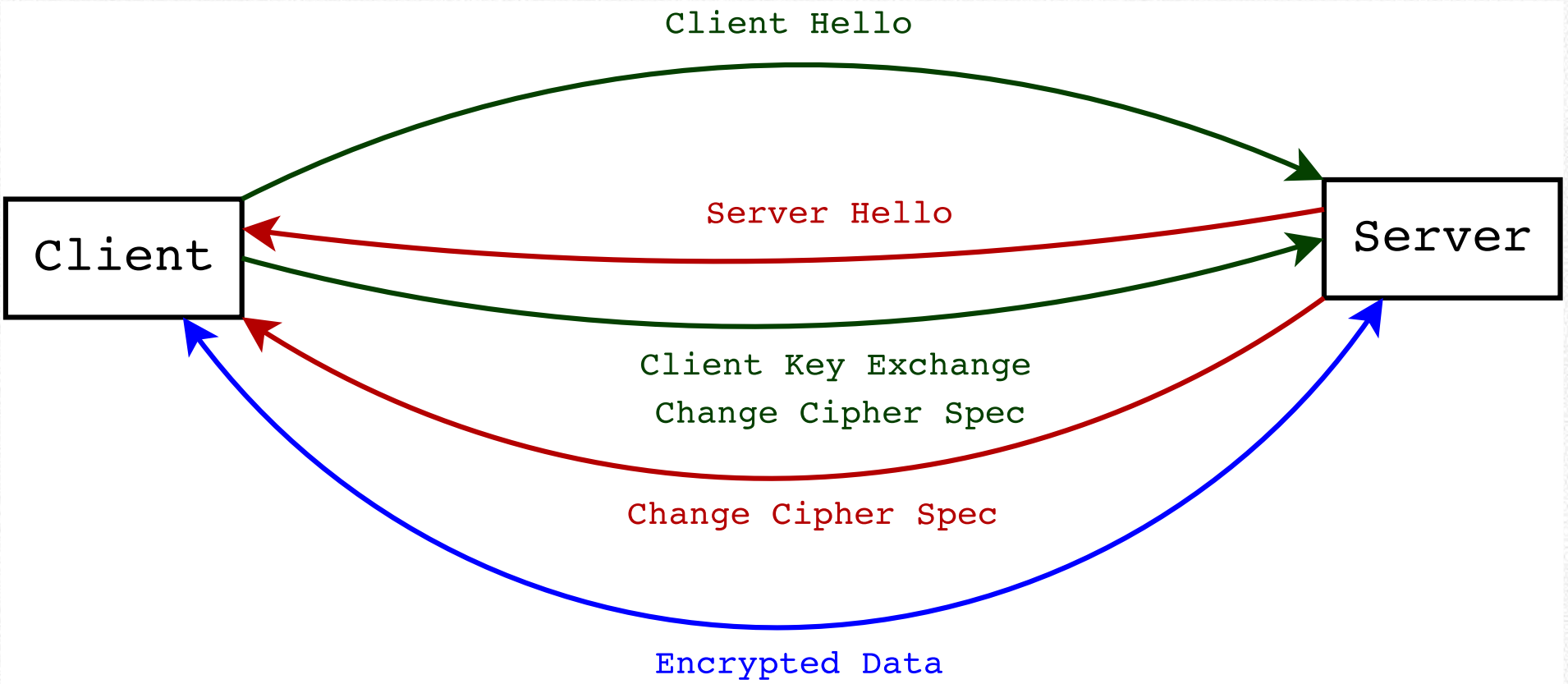
DEEPSEC

- Kürzerer Handshake (Illustration)
- Downgrade Protection
  - MAC über alle Handshake Messages
  - Nonce + DOWNGRD String in ServerHello.random
- Perfect Forward Secrecy für alle Sessions
  - Ausschaltung von Human-In-The-Middle
  - Detektieren von gefälschten Zertifikaten

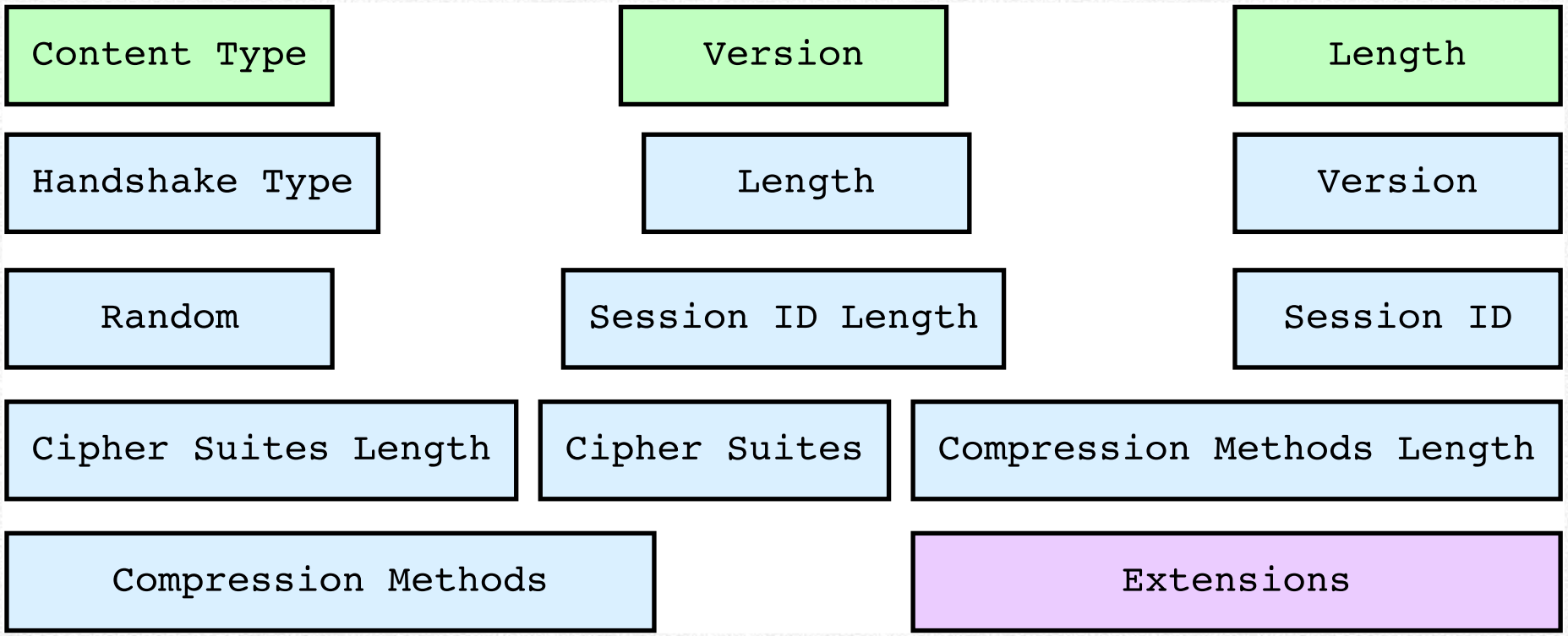
TLS v1.3 in OpenSSL 1.1.1

- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256
- TLS_AES_128_GCM_SHA256
- TLS_AES_128_CCM_8_SHA256
- TLS_AES_128_CCM_SHA256

TLS Verbindungsaufbau

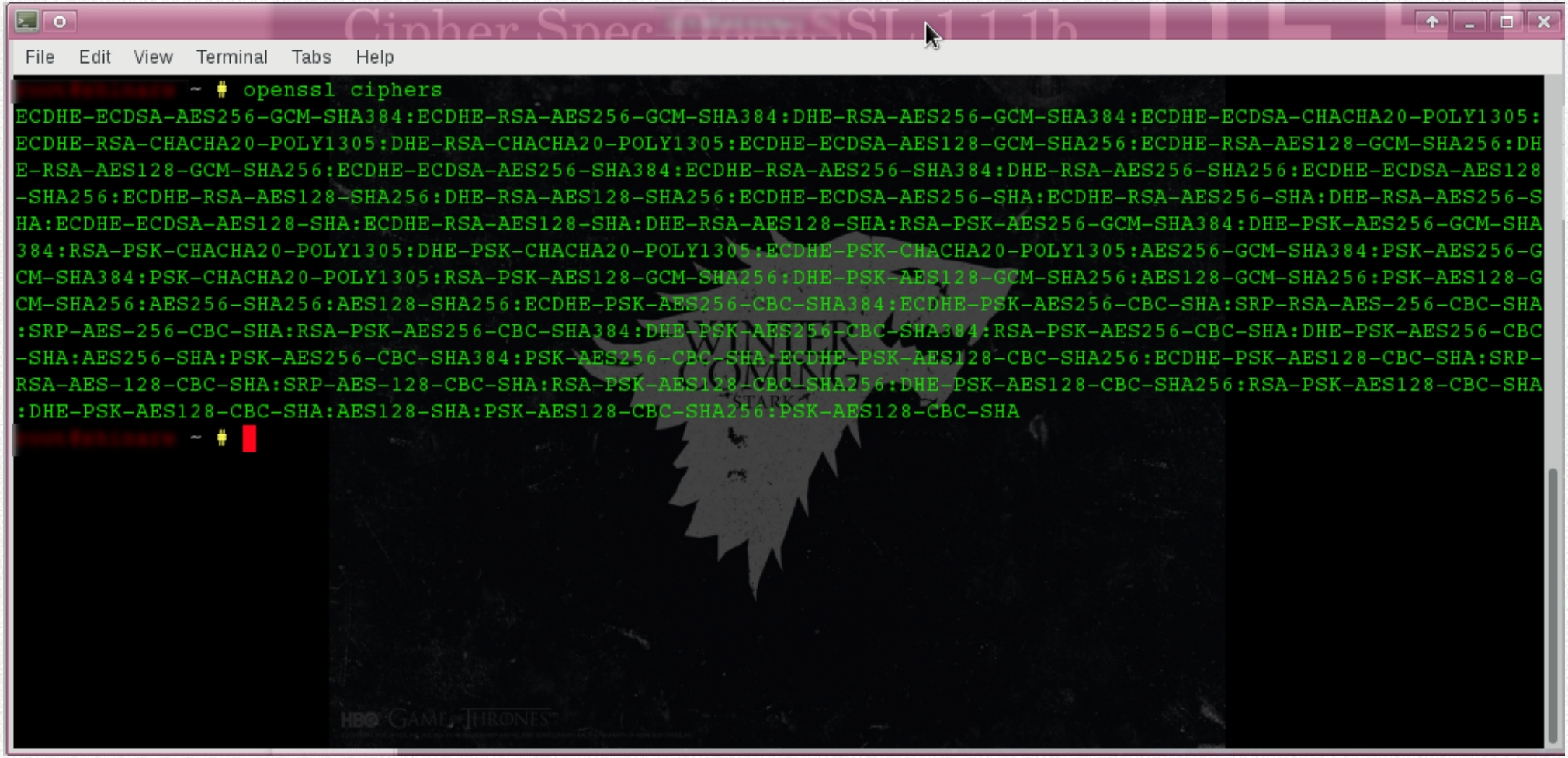


TLS Handshake Metadaten



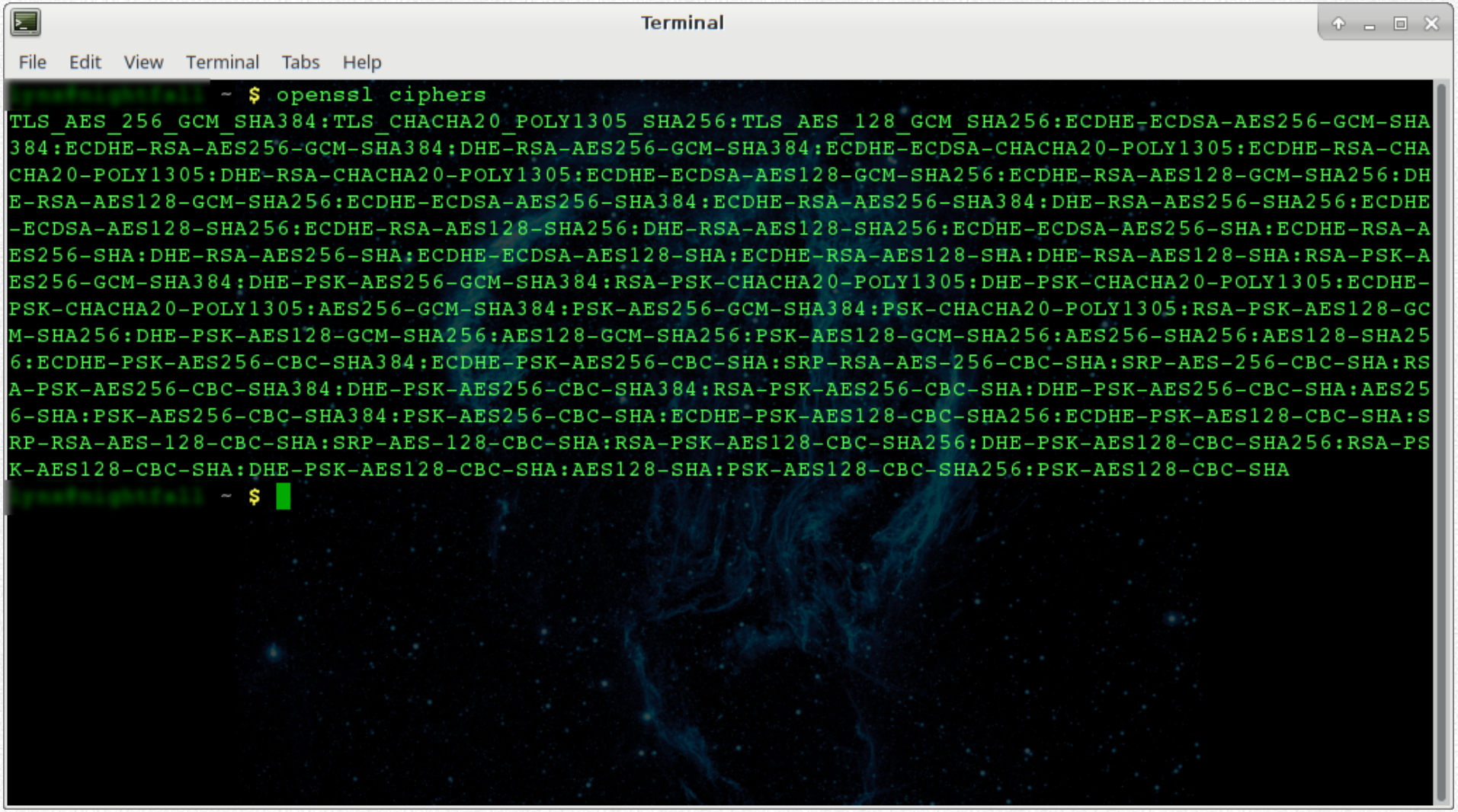
Cipher Spec OpenSSL 1.1.0j

DEEPSEC



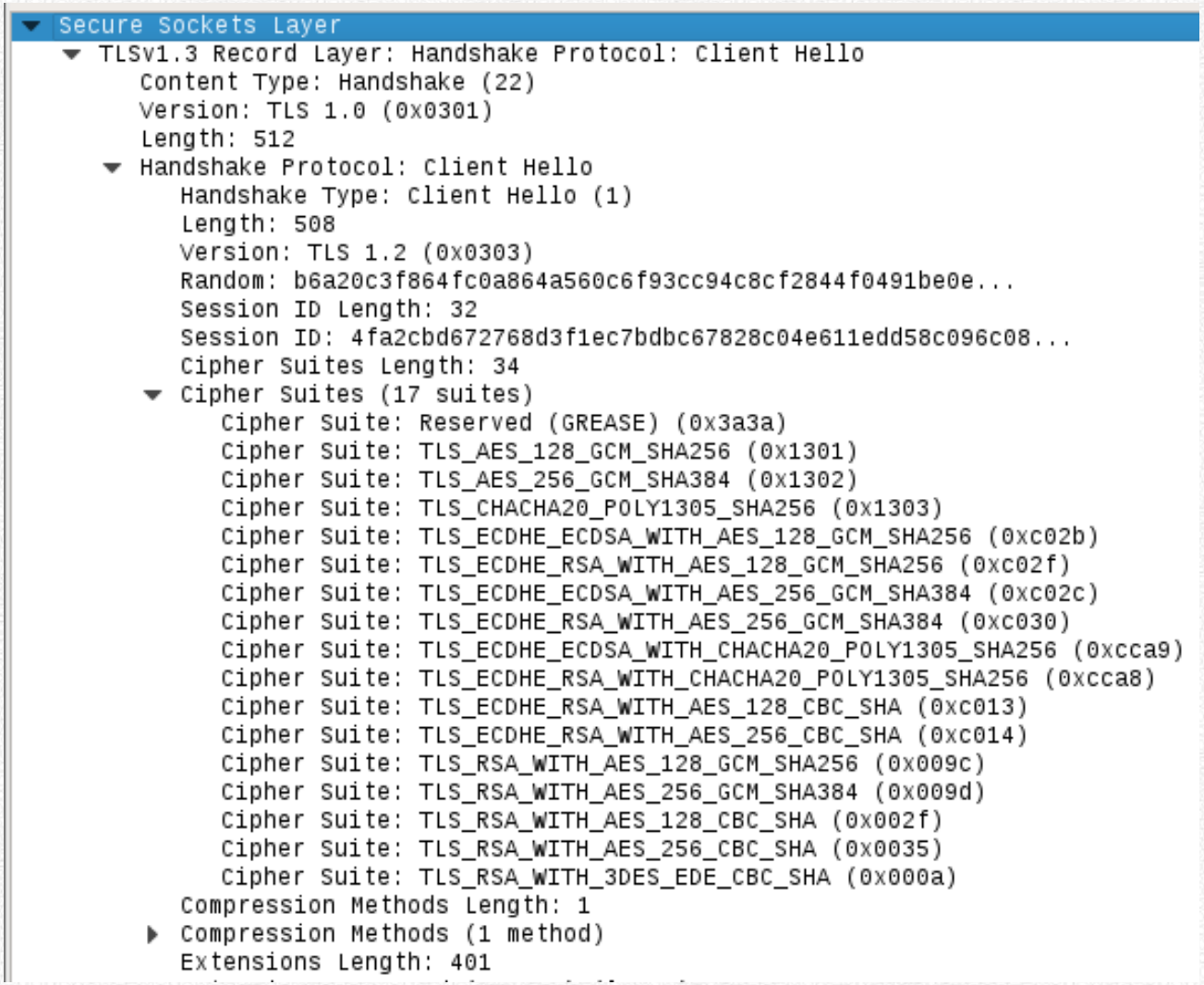
Cipher Spec OpenSSL 1.1.1b

DEEPSEC



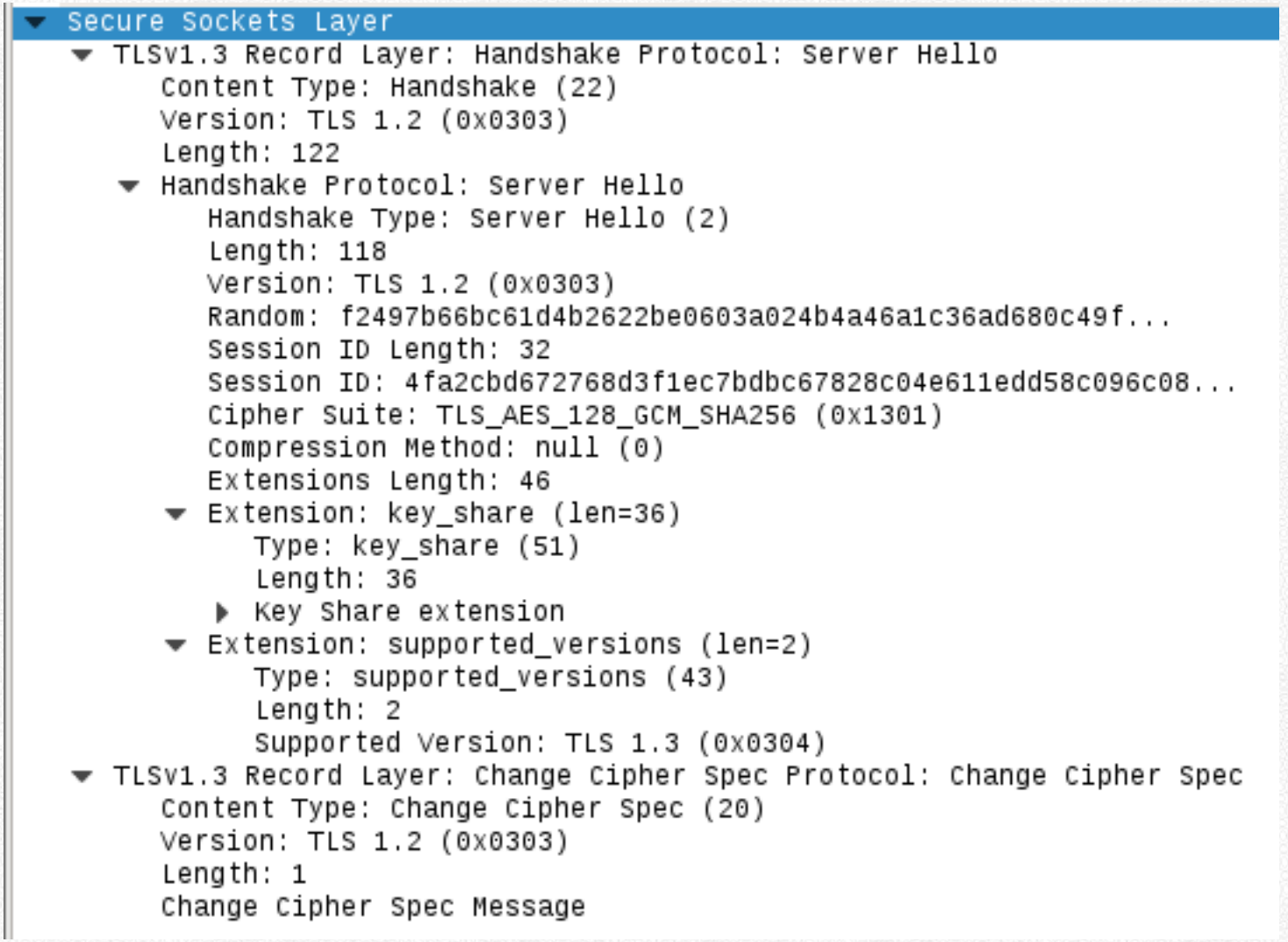
TLS Cipher Suites im Handshake

DEEPSEC



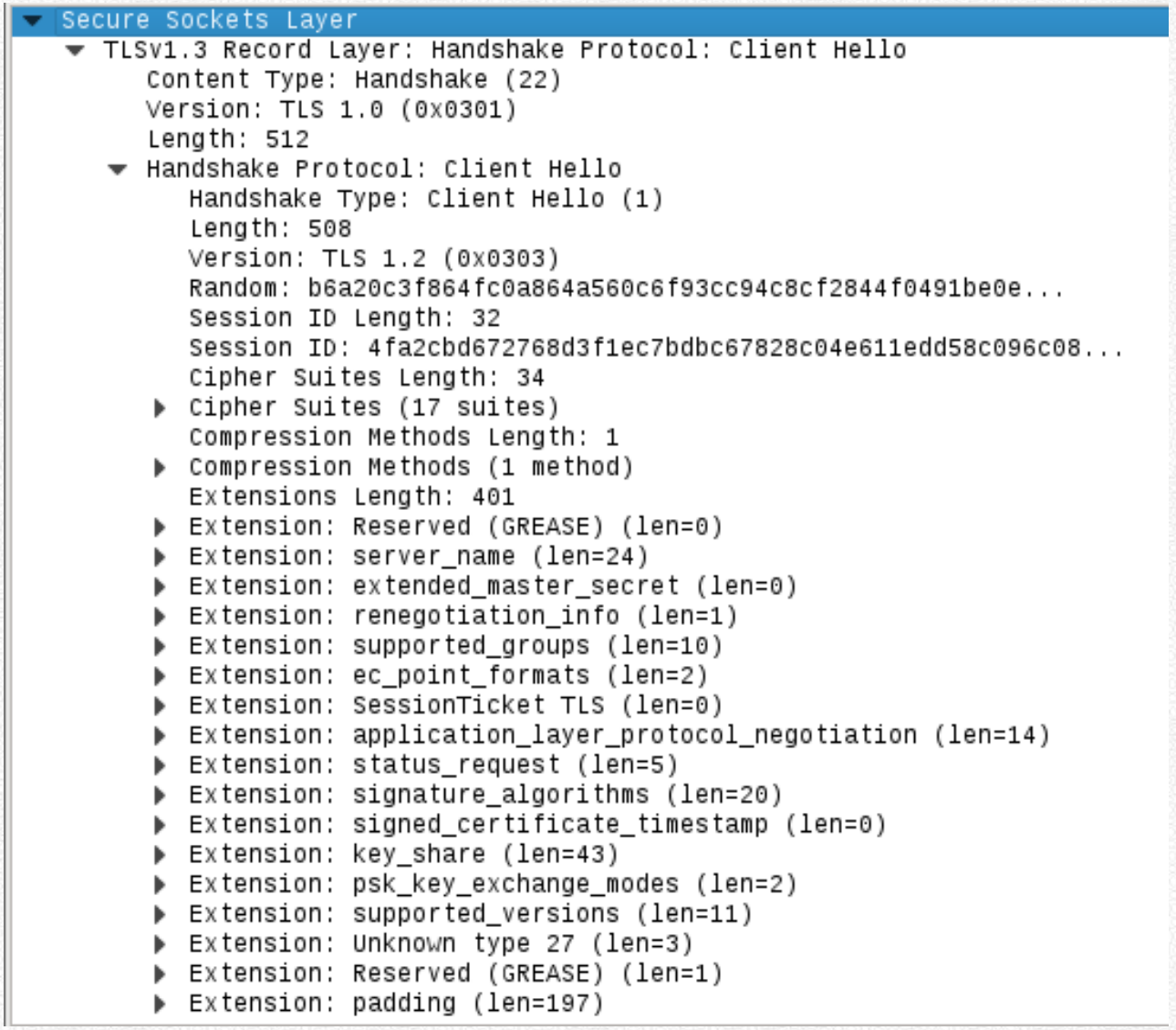
TLS Change Cipher Spec

DEEPSEC



TLS Extensions

DEEPSEC



TLS Extensions

DEEPSEC

- TLS Extensions sind Key-Value Paare
- Reihenfolge ist wichtig
  - Client-spezifisch
  - Server-spezifisch
- SSL/TLS Bibliotheken
  - Defaulteinstellungen
  - Applikationen können Parameter bestimmen / ändern

Exkurs: GREASE

- Generate Random Extensions And Sustain Extensibility
- Client erzeugt nicht definierte Extensions
- Test der TLS Version Intoleranz
- Aufspüren von defekten Implementationen
- Internet Draft (Vorschlag Google)
- Implementiert in Chrome 55+

Exkurs: GREASE

- GREASE Extension Codes bekannt
- TLS Fingerprint Algorithmus ignoriert GREASE Extensions
- Ausschnit aus JA3 (Fingerprinting) Python Code:

```
GREASE_TABLE = {0x0a0a: True, 0x1a1a: True, 0x2a2a: True, 0x3a3a: True,
                0x4a4a: True, 0x5a5a: True, 0x6a6a: True, 0x7a7a: True,
                0x8a8a: True, 0x9a9a: True, 0xaaaa: True, 0xbaba: True,
                0xcaca: True, 0xdada: True, 0xeaea: True, 0xfafa: True}
# GREASE_TABLE Ref: https://tools.ietf.org/html/draft-davidben-tls-grease-00
```

Fingerprinting?

DEEPSEC

- SSL/TLS Fingerprint ≠ TLS Fingerprinting
- kein Zusammenhang mit Fingerprint des Zertifikats,
- sondern…

TLS Fingerprinting - Vorarbeit

DEEPSEC

- SSL fingerprinting for p0f
- HTTP Client Fingerprinting Using SSL Handshake Analysis
  - Apache Modul
- TLS Fingerprinting with JA3 and JA3S
  - Varianten für Client und Server
  - Tests im Tor Netzwerk
  - Funktioniert mit TLS v1.3
- FingerPrinTLS Collection von Lee Brotherston

Methodik (JA3)

DEEPSEC

- Client Handshake – Extraktion von
  - TLSVersion
  - Ciphers
  - Extensions
  - EllipticCurves
  - EllipticCurvePointFormats
- Leere Extensions → mit 0 kodiert
- Wandlung in String: 769,47–53–5–10–49161…
- MD5 Hash

Methodik (JA3S)

DEEPSEC

- Server Handshake – Extraktion von
  - TLSVersion
  - Ciphers
  - Extensions
- Leere Extensions → mit 0 kodiert
- Wandlung in String: 769,47,65281–0–11–35–5–16…
- MD5 Hash

Verfügbare Datensätze

DEEPSEC

- 69 Malware Fingerprints (abuse.ch)
- Cisco (Blogartikel)
  - 4.000+ Fingerprints von „echten“ Netzwerken
  - 12.000+ Fingerprints aus Firmennetzwerken
  - 1.900+ Fingerprints publiziert
- ≅1.500 (JA3)
- 409 (FingerprinTLS)
- ≅ 1.684 (Kotzias et al.)

Projekt tlsfingerprint.io

DEEPSEC

- Use of TLS in Censorship Prevention (Studie)
- Verwendung echter Testdaten (University of Colorado Boulder)
  - 33.000 Studenten, 7.000 Angestellte
  - 9 Monate
  - 11 Milliarden TLS Verbindungen
  - kein UDP TLS
  - keine TLS Fragmente analysiert
  - keine out-of-order TCP Pakete analysiert
- Ergebnisse publiziert

Projekt tlsfingerprint.io

DEEPSEC

Rank	Version	Connections
1	TLS 1.2 (0x0303)	49.1%
2	TLS 1.3 (0x0304)	47.5%
3	TLS 1.0 (0x0301)	1.7%
4	TLS 1.3-draft-23 (0x7f17)	1.6%
5	TLS 1.3-draft-28 (0x7f1c)	0.1%
6	TLS 1.3-draft-18 (0x7f12)	0.0%
7	TLS 1.1 (0x0302)	0.0%
8	TLS 1.3-draft-26 (0x7f1a)	0.0%
9	Unknwon (0x7e02)	0.0%
10	SSL 3.0 (0x0300)	0.0%
11	Unknwon (0x7e01)	0.0%
12	Unknwon (0xfb28)	0.0%
13	TLS 1.3-draft-16 (0x7f10)	0.0%
14	Unknwon (0x7e03)	0.0%
15	TLS 1.3-draft-20 (0x7f14)	0.0%
16	TLS 1.3-draft-22 (0x7f16)	0.0%

Rank	Supported Version	Fingerprints	Connections
1	TLS 1.2 (0x0303)	555	48.3%
2	TLS 1.1 (0x0302)	482	48.0%
3	TLS 1.0 (0x0301)	463	48.0%
4	TLS 1.3 (0x0304)	395	47.5%
5	GREASE (0x0a0a)	152	30.6%
6	TLS 1.3-draft-23 (0x7f17)	75	1.6%
7	Unknown (0xfb1a)	24	1.0%
8	TLS 1.3-draft-28 (0x7f1c)	44	0.1%
9	TLS 1.3-draft-18 (0x7f12)	45	0.0%
10	TLS 1.3-draft-26 (0x7f1a)	18	0.0%
11	Unknown (0xfb17)	8	0.0%
12	Unknown (0x7e02)	7	0.0%
13	SSL 3.0 (0x0300)	10	0.0%
14	TLS 1.3-draft-27 (0x7f1b)	8	0.0%
15	Unknown (0x7e01)	6	0.0%
16	TLS 1.3-draft-19 (0x7f13)	1	0.0%
17	TLS 1.3-draft-20 (0x7f14)	1	0.0%
18	TLS 1.3-draft-21 (0x7f15)	1	0.0%
19	TLS 1.3-draft-22 (0x7f16)	1	0.0%
20	TLS 1.3-draft-24 (0x7f18)	1	0.0%
21	TLS 1.3-draft-25 (0x7f19)	1	0.0%

DEEPSEC

Projekt tlsfingerprint.io

DEEPSEC

Cipher Suite	Fingerprints	% Client Hellos	% Server Hellos
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)	5483	96.9%	0.9%
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)	5376	96.7%	1.0%
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)	4784	95.1%	12.8%
TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)	5663	95.1%	0.5%
TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)	5576	94.8%	1.1%
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)	4141	94.3%	1.6%
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)	4539	94.1%	43.2%
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)	3823	93.2%	17.5%
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)	4321	86.4%	0.3%
TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)	4254	86.1%	0.0%

Showing 1 to 10 of 423 entries

Previous 1 2 3 4 5 ... 43 Next

Projekt: tlsfingerprint.io

DEEPSEC

Signature Algorithm	Unique Fingerprints (%)	Connections
ecdsa_secp256r1_sha256 (0x0403)	9027 (76.0%)	98.2%
rsa_pkcs1_sha256 (0x0401)	9084 (76.5%)	97.3%
rsa_pkcs1_sha1 (0x0201)	8981 (75.7%)	97.3%
rsa_pkcs1_sha384 (0x0501)	8989 (75.7%)	97.2%
ecdsa_secp384r1_sha384 (0x0503)	8882 (74.8%)	96.9%
rsa_pkcs1_sha512 (0x0601)	8611 (72.5%)	94.6%
rsa_pss_rsae_sha256 (0x0804)	1872 (15.8%)	61.2%
rsa_pss_rsae_sha512 (0x0806)	1822 (15.3%)	60.2%
rsa_pss_rsae_sha384 (0x0805)	1821 (15.3%)	60.2%
ecdsa_sha1 (0x0203)	7835 (66.0%)	59.2%

Showing 1 to 10 of 44 entries

Previous 1 2 3 4 5 Next

Projekt: tlsfingerprint.io

DEEPSEC

- Genauigkeit der Fingerprints
  - Handshake Messages sind variabel
  - Server senden mehr Variationen
- tlsfingerprint.io: populärste IP Adresse (Google Server)
  - 199 eindeutige Fingerprints als Antwort auf
  - 1.494 Client Hello Fingerprints
- häufigstes Client Hello bekam 750 Server Hello Fingerprints

Implementationen (JA3)

DEEPSEC

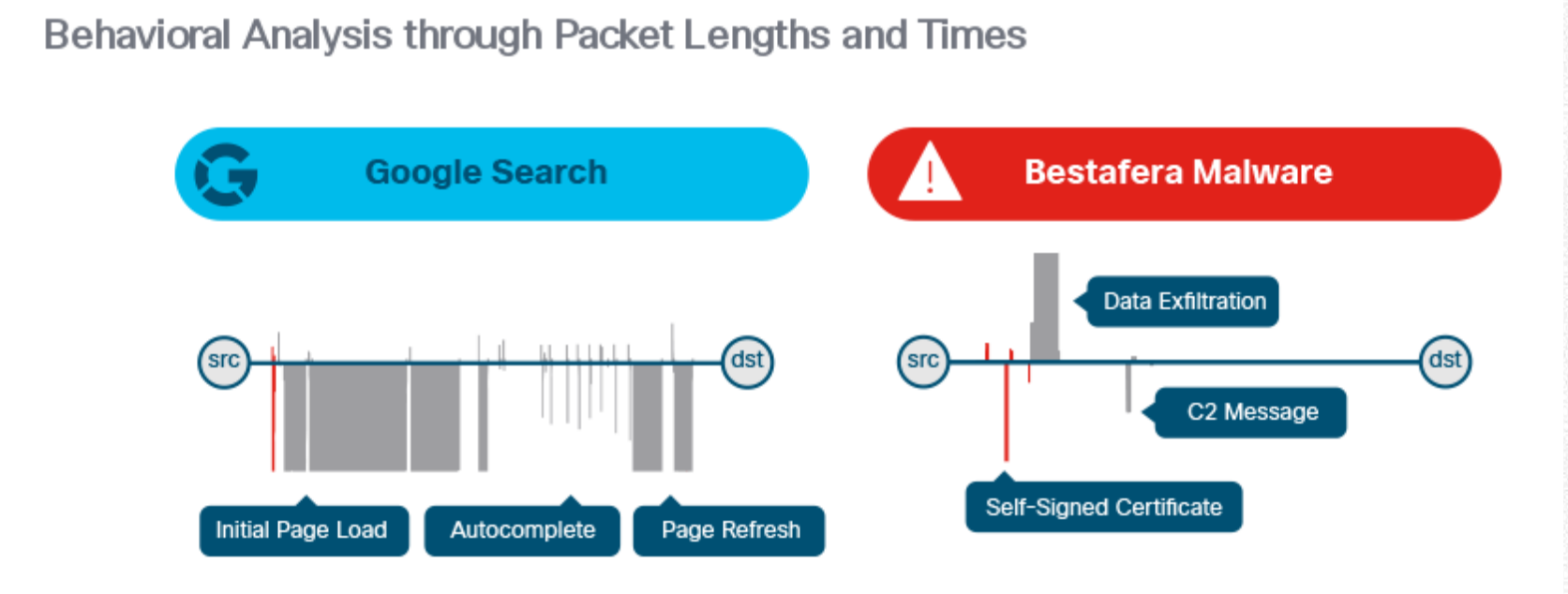
- Bro/Zeek
- Go/Java/Python
- Darktrace
- Suricata
- Splunk
- MantisNet
- Moloch
- Trisul NSM
- MISP
- Elastic.co Packetbeat
- ICEBRG
- Redsocks
- Netwitness
- ExtraHop
- Security Onion
- …

- uTLS Bibliothek
- Fork der Go crypto/tls Bibliothek
- Gegenmaßnahmen zu Fingerprinting
  - komplette Kontrolle des Handshakes
  - Zugriff auf alle Bits der Client Hello Message
- Imitieren von Handshakes
  - “Parroting”
  - nur für den Handshake

Erweiterung: C&C Analyse

DEEPSEC

- Verwendung von TLS bei Schadsoftware
  - Infizierung
  - C&C
  - Nachladen von Code
- TLS leicht erkennbar
  - Handshake
  - Non-standard Ports verbergen nichts
- Applikation verwendet Komponenten



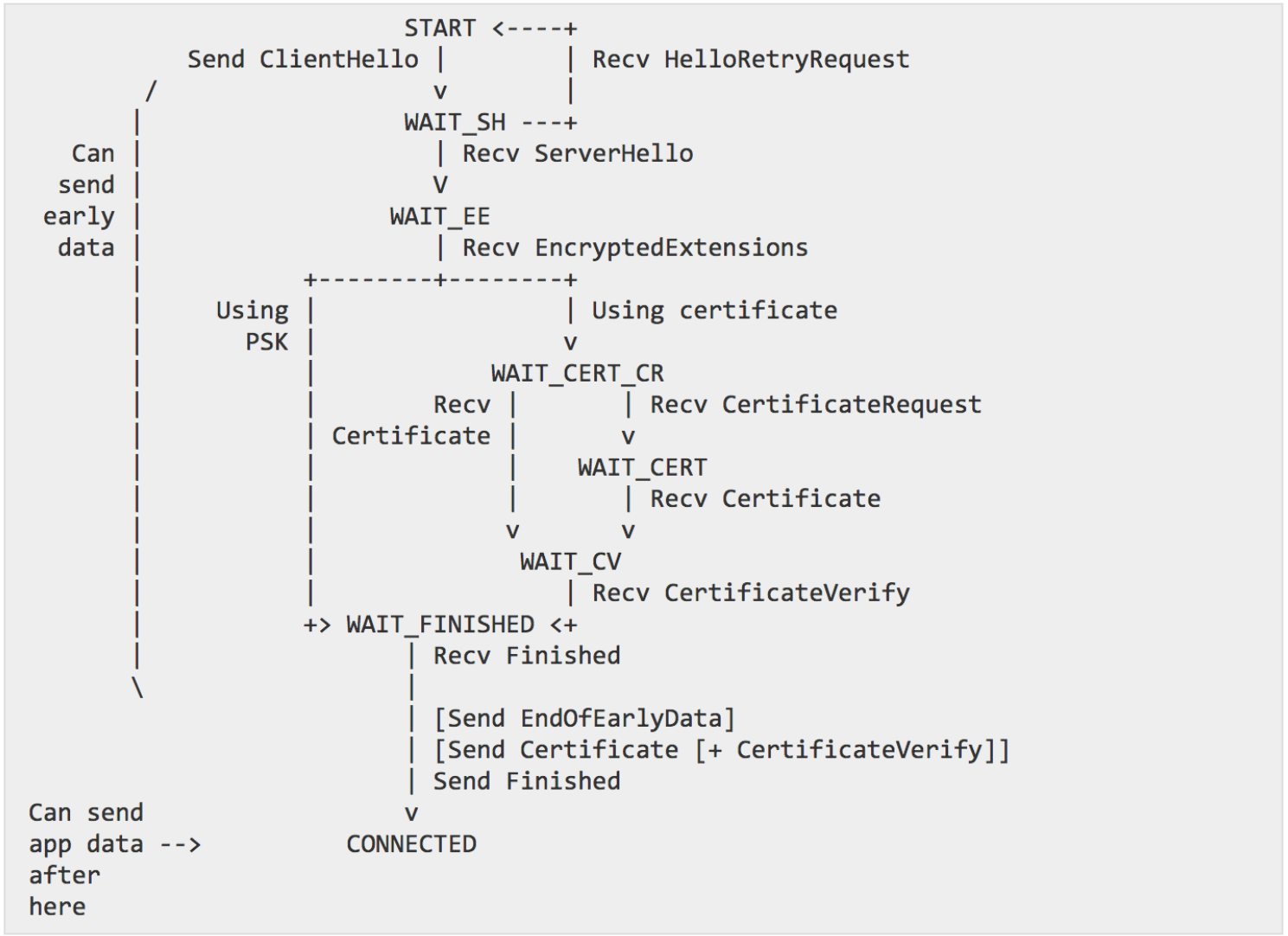
Quelle: Detecting Encrypted Malware Traffic (Without Decryption)

Zusammenfassung

DEEPSEC

- TLS Fingerprinting vielversprechend
- leicht zu verwenden
- mehrere Projekte, bereits viele Implementationen
- Datenbanken mit Signaturen erforderlich
  - einige vorhanden
  - ideales Community Projekt (siehe DSHIELD)
  - Referenzsignaturen hilfreich
- sehr nützlich für Debugging (z.B. „Secure“/Secure Messenger)

Fragen?


START <----+
Send ClientHello | | Recv HelloRetryRequest
v |
WAIT_SH ---+
Can | | Recv ServerHello
send | V
early | WAIT_EE
data | | Recv EncryptedExtensions
+--------+--------+
Using | | Using certificate
PSK | v
WAIT_CERT_CR
Recv | | Recv CertificateRequest
Certificate | v
| WAIT_CERT
| | Recv Certificate
v v
WAIT_CV
| Recv CertificateVerify
+> WAIT_FINISHED <+
| Recv Finished
|
| [Send EndOfEarlyData]
| [Send Certificate [+ CertificateVerify]]
| Send Finished
Can send v
app data --> CONNECTED
after
here


DEEPSEC

Kontaktmöglichkeiten

DEEPSEC

- ✉ rpfeiffer@deepsec.net
- 🔐 0x518A0576C3A9FF76 (PGP/GnuPG)
- FP: AE26 3866 FB54 4A5E 0BE5 AD90 8531 093E 6E40 37AF
- 📞 +43.676.5626390 (Signal verfügbar/empfohlen)
- 📞 +807 949 050 59 (GSMK Cryptophone™)
- 🖝 9EKKN34F (Threema)

Weiterführende Links

DEEPSEC

- TLSFingerprint.io
- TLS Fingerprinting (Lee Brotherston)
- TLS Fingerprinting JA3 / JA3S (Salesforce.com)
- TLS Fingerprinting (Cisco)
- TLS Fingerprinting Implementation (Cisco)
- HTTP client fingerprinting using SSL handshake analysis
- JA3 SSL Fingerprint Webseite
- SSL Blacklist (SSLBL) von abuse.ch
- TLS Fingerprint Collection (University of Colorado Boulder)

Quellen

DEEPSEC

- Titelbild aus Artikel

  What the President Could Do If He Declares a State of Emergency

  von Elizabeth Goitein

Exkurs: Middleboxen

DEEPSEC

- TLS – Inhalte können nicht inspiziert werden
- Firewalls / Application Layer Gateways
  - führen eigene (interne) Certificate Authority
  - Clients kennen eigene CA (durch Import)
  - Generierung von Zertifikaten in Echtzeit
- Entwurf für interne Netzwerke
- Öffentliche CAs bieten Sub-CAs für Filter an
  - stark kritisiert
  - exponiert durch Cert Pinning & CA Transparency

- Middle Box Problematik
- Studie The Security Impact of HTTPS Interception

Product	Grade	Validates Certificates	Modern Ciphers	Advertises RC4	TLS Version	Grading Notes
A10 vThunder SSL Insight	F	✓	✓	Yes	1.2	Advertises export ciphers
Blue Coat ProxySG 6642	A*	✓	✓	No	1.2	Mirrors client ciphers
Barracuda 610Vx Web Filter	C	✓	✗	Yes	1.0	Vulnerable to Logjam attack
Checkpoint Threat Prevention	F	✓	✗	Yes	1.0	Allows expired certificates
Cisco IronPort Web Security	F	✓	✓	Yes	1.2	Advertises export ciphers
Forcepoint TRITON AP-WEB Cloud	C	✓	✓	No	1.2	Accepts RC4 ciphers
Fortinet FortiGate 5.4.0	C	✓	✓	No	1.2	Vulnerable to Logjam attack
Juniper SRX Forward SSL Proxy	C	✓	✗	Yes	1.2	Advertises RC4 ciphers
Microsoft Threat Mgmt. Gateway	F	✗	✗	Yes	SSLv2	No certificate validation
Sophos SSL Inspection	C	✓	✓	Yes	1.2	Advertises RC4 ciphers
Untangle NG Firewall	C	✓	✗	Yes	1.2	Advertises RC4 ciphers
WebTitan Gateway	F	✗	✓	Yes	1.2	Broken certificate validation

Fig. 3: **Security of TLS Interception Middleboxes**—We evaluate popular network middleboxes that act as TLS interception proxies. We find that nearly all reduce connection security and five introduce severe vulnerabilities. *Mirrors browser ciphers.

Exkurs: EU – ETSI ETS

- Einwand von BITS auf IETF Mailing Liste
- ETSI ETS = TLS v1.3 – PFS
- „Data Centre TLS“

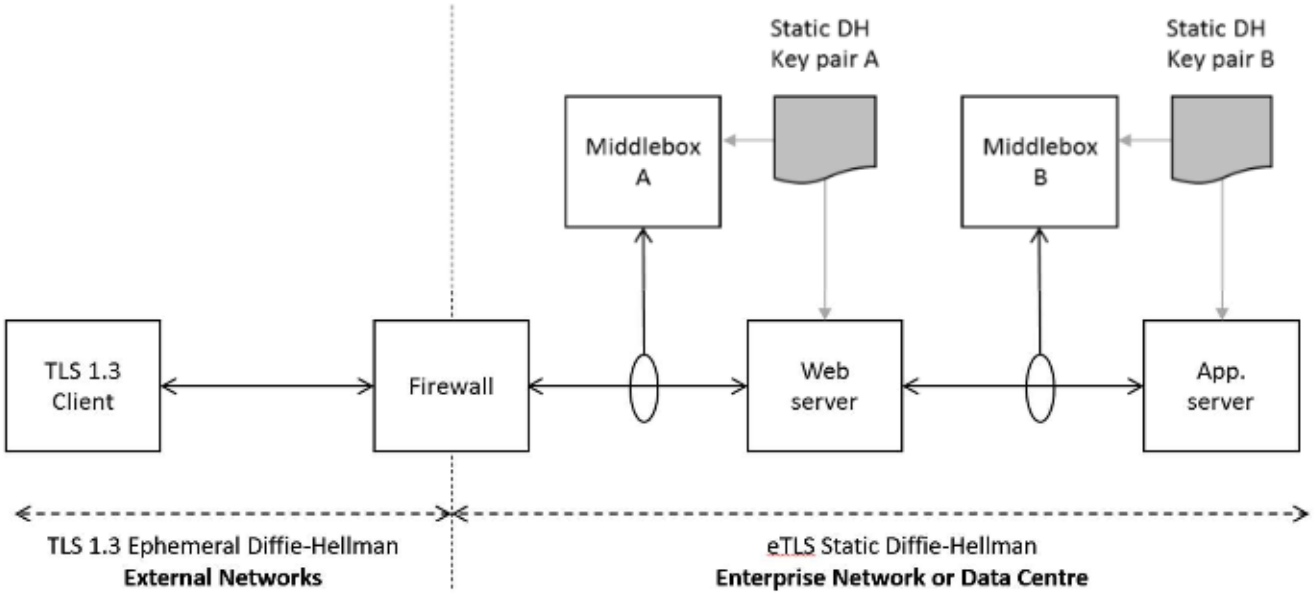


Figure 4.1: eTLS architecture with enterprise servers

Exkurs: USA – NIST Empfehlungen

October 15, 2018

Transport Layer Security (TLS) provides mechanisms for protecting data during electronic dissemination across the Internet. **Draft NIST Special Publication (SP) 800-52 Rev.2, *Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations***, provides guidance for selecting and configuring TLS protocol implementations using NIST-recommended cryptographic algorithms and Federal Information Processing Standards (FIPS). The document requires that government TLS servers and clients support TLS 1.2 configured with FIPS-based cipher suites.

This second draft extends the deadline by which agencies are urged to support TLS 1.3 to January 1, 2024. Moreover, it clarifies that TLS 1.3 is intended to coexist with TLS 1.2 rather than replace it. An appendix has also been added to discuss key exchange using RSA key transport and includes a list of cipher suites that may be used if a transition period is needed. The extensions guidance now clarifies which versions of TLS each extension applies to and provides guidance on the raw public keys extension.

A public comment period for this document **is open until November 16, 2018**.

Quelle: Second Draft of NIST's Transport Layer Security (TLS) Guidance

Exkurs: ETSI Warnungen

- Europäische Standards-Organisation warnt USA vor TLS 1.3
- CVE-2019-9191 – fehlende PFS in ETSI ETS
  - CVSS v3.0 Medium (AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:N/A:N)
  - CVSS v2.0 Medium (AV:N/AC:M/Au:N/C:P/I:N/A:N)
- PCI 3.1 empfiehlt TLS v1.2 oder besser