Downgrade Resilience in Key Exchange

markulf kohlweiss

joint work with:
k. bhargavan, c. brzuska, c. fournert,
m. green, s. zanella-beguelin
Downgrade as an everyday phenomenon

https://

http://
## TLS protocol suite – not a single protocol

### Hello Messages

- **Client**
- **Server**

### Negotiation
- **Client**
- **Server**
- Hello Messages

- **Client**
- **Server**
- 4 protocol versions
- 100s of ciphersuites
- 10s of extensions

### Key Exchange
- **Client**
- **Server**

- **Client**
- **Server**
- RSA key transport
- DHE/ECDHE with RSA/DSA/ECDSA signatures
- PSK, SRP, ...

### Confirmation
- **Client**
- **Server**

- **Client**
- **Server**
- Finished Messages
- MAC based on MD5+SHA1, SHA1, SHA256

### Data
- **Client**
- **Server**

- **Client**
- **Server**
- HMAC with AES-CBC
- HMAC with RC4
- AES-GCM
Crypto failures

Protocol weaknesses

Implementation bugs

MD5

Renegotiation Attack

BEAST (Rogaway 02)

ECDHE Cross-protocol Attack

RC4

RSA 512 bit

SHA1

SLOTH

DROWN

OpenSSL entropy

CRIME

Lucky13

POODLE

EarlyCCS

Heartbleed

SKIP

Logjam

FREAK

Renegotiation Attack

ECDHE Cross-protocol Attack

BEAST (Rogaway 02)

RC4

RSA 512 bit

SHA1

SLOTH

DROWN

OpenSSL entropy

CRIME

Lucky13

POODLE

EarlyCCS

Heartbleed

SKIP

Logjam

FREAK

Renegotiation Attack

ECDHE Cross-protocol Attack

BEAST (Rogaway 02)

RC4

RSA 512 bit

SHA1

SLOTH

DROWN

OpenSSL entropy

CRIME

Lucky13

POODLE

EarlyCCS

Heartbleed

SKIP

Logjam

FREAK
- POODLE
- LOGJAM
- SLOTH
Our contribution

1. Definition that tolerate weak algorithms
   – and capture downgrade attacks

2. Modular proof strategy

   • Analyse downgrade security of SSH, IKE, ZRTP, TLS

   • Prove downgrade security for SSH and TLS 1.3

   New countermeasures designed together with core-design team of TLS 1.3
Negotiation

- **Inputs:**
  - $\text{config}_C \& \text{config}_S$: supported versions, ciphers, groups, long-term keys

- **Outputs:**
  - $\text{mode}$: negotiated version, cipher, group, etc.

- **Ideal negotiation:**
  - $\text{mode} = \text{Nego} (\text{config}_C, \text{config}_S)$
Transcript authentication vs. Downgrades

- **Authentication**
  
  If my negotiated mode uses only strong algorithms, then my partner and I agree on keys, identities and mode.

- Authentication does not guarantee negotiation of a strong mode.
  
  - **Intersection** of config\textsubscript{C} & config\textsubscript{S} must be strong!
  - What if config\textsubscript{C} & config\textsubscript{S} include a legacy algorithm?
  - What are minimal requirements on config\textsubscript{C} & config\textsubscript{S}?
POODLE

Negotiation

Client

Hello Messages

Server

- 4 protocol versions
- 100s of ciphersuites
- 10s of extensions

Key Exchange

- RSA key transport
- DHE/ECDHE with RSA/DSA/ECDSA signatures
- PSK, SRP, ...

Confimation

- MAC based on MD5+SHA1, SHA1, SHA256

Data encryption

- HMAC with AES-CBC
- HMAC with RC4
- AES-GCM

[Dowling and Stebila 2015]
LOGJAM

Client C

Knows $sk_C, pk_S$

$conf_{G_{2048}, G_{512}}$

$[G_{2048}, G_{512}]$

$[G_{512}]$

$m_1 = g^x \mod p_{512}$

$m_2 = g^y \mod p_{512}$

sign($sk_S$, hash($m1||m2$))

$k = kdf(g^{xy} \mod p_{512})$

mac($k$, transcript)

MitM

Server S

Knows $sk_S, pk_C$

$conf_{G_{2048}, G_{512}}$

$G_{512}$

sign($sk_S$, transcript')

$k = kdf(g^{xy} \mod p_{512})$

mac($k$, transcript')
<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Negotiation</strong></td>
<td>• 4 protocol versions</td>
</tr>
<tr>
<td></td>
<td>• 100s of ciphersuites</td>
</tr>
<tr>
<td></td>
<td>• 100s of extensions</td>
</tr>
<tr>
<td><strong>Key Exchange</strong></td>
<td>• Key transport</td>
</tr>
<tr>
<td></td>
<td>• DHE/ECDHE with RSA/DSA/ECDSA signatures</td>
</tr>
<tr>
<td></td>
<td>• PSK, SRP, ...</td>
</tr>
<tr>
<td><strong>Confirmation</strong></td>
<td>• MAC based on MD5+SHA1, SHA1, SHA256</td>
</tr>
<tr>
<td><strong>Data encryption</strong></td>
<td>• HMAC with AES-CBC</td>
</tr>
<tr>
<td></td>
<td>• HMAC with RC4</td>
</tr>
<tr>
<td></td>
<td>• AES-GCM</td>
</tr>
<tr>
<td>Client</td>
<td>Server</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
</tr>
</tbody>
</table>
| **Negotiation** | • 4 protocol versions  
• 100s of ciphersuites  
• Protocol extensions |
| **Key Exchange** | • Key transport  
• DHE/ECDSA  
• RSA/DSA/signatures  
• PSK, SRP, ... |
| **Confirmation** | • MAC based on MD5+SHA1, SHA1, SHA256 |
| **Data encryption** | • HMAC with AES-CBC  
• HMAC with RC4  
• AES-GCM |

\[
\text{md5}(m_1 \parallel m'_2) = \text{md5}(m'_1 \parallel m_2)
\]
Downgrade secure configurations

• Downgrade protection (DP) only if
  – $\text{config}_C$ requires good public keys and signatures scheme
  – $\text{config}_S$ has preference for downgrade secure version

• Clients and servers interoperate with everyone; get desired mode only when $\text{DP}(\text{config}_C, \text{config}_S)$. 
Protocol execution model

Adversary controls generation of keys and sessions

**Configurations:**
- algorithms and keys supported by sessions

Sessions assign **variables**
Downgrade security

\[
\text{config} := \text{config}_C \\
\text{uid} := \text{uid} \\
\text{mode} := \text{mode} \\
\text{complete} := \text{true}
\]

\[
\text{c} \\
\text{c} \\
\text{s} \\
\text{s}
\]

\[
\text{DP}(C.\text{config}, S.\text{config}) \text{ but} \\
\text{mode} \neq \text{Nego}(C.\text{config}, S.\text{config})
\]

What if server does not exist?
Our contribution

1. Definition that tolerate weak algorithms – and capture downgrade attacks

2. Modular proof strategy

   • Analyse downgrade security of SSH, IKE, ZRTP, TLS

   • Prove downgrade security for SSH and **TLS 1.3**

   **NEW!** New countermeasures designed together with core-design team of TLS 1.3
Reducing complex real-world protocol analysis ...

draft-ietf-tls-tls13-latest

Abstract
This document specifies Version 1.3 of the Transport Layer Security (TLS) protocol. The TLS protocol allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 23, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents
... using simulation ...
... into analysis of downgrade sub-protocol (TLS 1.3)

Client C
- Initialized with $config_C$
- $m_0 = (n_C, F_0(config_C))$
- $m_0' = G_S$
- $m_1 = (n_C, F_1(config_C))$

Server S
- Initialized with $config_S$
- $uid = (n_C, n_S)$
- $mode = nego(F_1(config_C), config_S)$
  $= (v, a_S, G_S, pk_S, hash_1)$
- $m_2 = (n_S, v, a_S, G_S, pk_S)$
- $sign(sk_S, hash_1(H(m_1, m_2, -)))$
- $check(config_C, mode)$
- $complete = true$

Server signs full transcript with strong signature and hash algorithms?
Server S

Client C

- \( m_0 = (n_C, F_0(config_C)) \)
- \( m_0' = G_S \)
- \( m_1 = (n_C, F_1(config_C)) \)
- \( m_2 = (n'_S, v, a_S, G_S, pk_S) \)
- \( \text{sign}(sk_S, hash_1(H(m_0, m_0', m_1, m_2, -))) \)

- \( \text{uid} = (n_C, n_S) \)
- \( \text{mode} = (v, a_S, G_S, pk_S, hash_1) \)
- \( \text{check}(config_C, \text{mode}) \)
- \( \text{complete} = \text{true} \)

- \( \text{uid} = (n_C, n_S); n'_S = n_S || \text{maxv}(config_S) \)
- \( \text{mode} = \text{nego}(F_1(config_C), config_S) = (v, a_S, G_S, pk_S, hash_1) \)

- \( \text{complete} = \text{true} \)
Downgrade security of TLS 1.3

• **Good news:**
  TLS 1.3 now has secure downgrade sub-protocol
  – **nonce and signatures:** unique server signs all network input to *nego* and result.

• **What do we do about version downgrade?**
  – Can an attacker downgrade TLS 1.3 to TLS 1.2 and remount Logjam?
Version downgrade resilience

• TLS 1.3 server signatures cover versions
  But TLS 1.2 signatures do not cover the version

• How do we patch TLS 1.2 to prevent downgrades?
  – Finished messages cannot help
  – Look away: put max server version in server nonce signed in all versions of TLS

• Good news: $\text{DP}(\text{config}_C, \text{config}_S)$ for TLS 1.0-1.3 if
  – countermeasure implemented
  – no RSA key transport
Downgrade Resilience in Key Exchange

https://www.mitls.org/

https://eprint.iacr.org/2016/072