Improved KRACK Attacks Against WPA2 Implementations

Mathy Vanhoef — @vandoefm
OPCDE, Dubai, 7 April 2018
Overview

Key reinstalls in 4-way handshake

Practical impact

New KRACKs

Lessons learned
Overview

Key reinstalls in 4-way handshake

Practical impact

New KRACKs

Lessons learned
The 4-way handshake

Used to connect to any protected Wi-Fi network
› Provides mutual authentication
› Negotiates fresh PTK: pairwise transient key

Appeared to be secure:
› No attacks in over a decade (apart from password guessing)
› Proven that negotiated key (PTK) is secret\(^1\)
› And encryption protocol proven secure\(^5\)
4-way handshake (simplified)

optional 802.1x authentication
4-way handshake (simplified)

PTK = Combine(shared secret, ANonce, SNonce)
4-way handshake (simplified)

Attack isn’t about ANonce or SNonce reuse

PTK = Combine(shared secret, ANonce, SNonce)
4-way handshake (simplified)
4-way handshake (simplified)

optional 802.1x authentication →

\[\text{Msg1}(r, \text{ANonce})\]

\[\text{Derive PTK}\]

\[\text{Msg2}(r, \text{SNonce})\]

\[\text{Derive PTK}\]

\[\text{Msg3}(r+1; \text{GTK})\]

\[\text{Install PTK & GTK}\]

\[\text{Msg4}(r+1)\]

\[\text{Install PTK}\]
4-way handshake (simplified)

optional 802.1x authentication

Msg1(r, ANonce)

Derive PTK

Msg2(r, SNonce)

P TK is installed

Msg3(r+1; GTK)

Msg4(r+1)

Install PTK & GTK

Install PTK
4-way handshake (simplified)

optional 802.1x authentication

Msg1(r, ANonce)

Derive PTK

Msg2(r, SNonce)

Derive PTK

Msg3(r+1; GTK)

Msg4(r+1)

Install PTK & GTK

Install PTK

encrypted data frames can now be exchanged
Frame encryption (simplified)

PTK \rightarrow \text{Mix} \rightarrow \text{Packet key}

\text{Nonce} \quad \text{(packet number)}

\text{Plaintext data} + \text{Keystream} \rightarrow \text{Encrypted data}

\rightarrow \text{Nonce reuse implies keystream reuse (in all WPA2 ciphers)}
4-way handshake (simplified)

optional 802.1x authentication

Msg1(r, ANonce)

Derive PTK

Msg2(r, SNonce)

Installing PTK initializes nonce to zero

Install PTK & GTK

Install PTK

encrypted data frames can now be exchanged
Reinstallation Attack

Channel 1

Channel 6
Reinstallation Attack

- Optional 802.1x authentication

<table>
<thead>
<tr>
<th>Phone</th>
<th>Router</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Msg1(r, ANonce)</strong></td>
<td><strong>Msg1(r, ANonce)</strong></td>
</tr>
<tr>
<td><strong>Msg2(r, SNonce)</strong></td>
<td><strong>Msg2(r, SNonce)</strong></td>
</tr>
<tr>
<td><strong>Msg3(r+1; GTK)</strong></td>
<td><strong>Msg3(r+1; GTK)</strong></td>
</tr>
</tbody>
</table>
Reinstallation Attack

optional 802.1x authentication

Msg1(r, ANonce)  Msg1(r, ANonce)
Msg2(r, SNonce)  Msg2(r, SNonce)
Msg3(r+1; GTK)  Msg3(r+1; GTK)
Msg4(r+1)  Block Msg4

Install PTK & GTK
Reinstallation Attack

Install PTK & GTK

\[ \text{Enc}_{\text{ptk}}^1 \{ \text{Msg4}(r+2) \} \]

\[ \text{Msg3}(r+2; \text{GTK}) \]

\[ \text{Msg4}(r+1) \]
Reinstallation Attack

In practice, $\text{Msg}_4$ is sent encrypted.
Reinstallation Attack

Key reinstallati{}on!  Nonce is reset
Reinstallation Attack

Same nonce is used!
Reinstallation Attack

Install PTK & GTK

Msg3(r+2; GTK)

Enc$_{ptk}^1 \{ \text{Msg4(r+2)} \}

Reinstall PTK & GTK

Enc$_{ptk}^1 \{ \text{Data(...) \}}$

Keystream
Reinstallation Attack

Install PTK & GTK

Enc$^1_{ptk}$ \{ Msg4(r+2) \}

Reinstall PTK & GTK

Enc$^1_{ptk}$ \{ Data(….) \}

Keystream

Msg4(r+1)

Msg3(r+2; GTK)

Enc$^1_{ptk}$ \{ Data(….) \}

Decrypted!
Overview

Key reinstall in 4-way handshake

Practical impact

New KRACKs

Lessons learned
General impact

- Transmit nonce reset
- Decrypt frames sent by victim
- Receive replay counter reset
- Replay frames towards victim
Cipher suite specific

AES-CCMP:
› No practical frame forging attacks

WPA-TKIP:
› Recover Message Integrity Check key from plaintext\textsuperscript{2,3}
› **Forge/inject** frames sent by the device under attack
Handshake specific

Group key handshake:
› Client is attacked, but only AP sends real broadcast frames
› Can only replay broadcast frames to client

4-way handshake:
› Client is attacked → replay/decrypt/forge
Implementation specific

iOS 10 and Windows: 4-way handshake not affected
  › Cannot decrypt unicast traffic (nor replay/decrypt)
  › But group key handshake is affected (replay broadcast)
  › Note: iOS 11 does have vulnerable 4-way handshake

wpa_supplicant 2.4+
  › Client used on Linux and Android 6.0+
  › On retransmitted msg3 will install all-zero key
Overview

Key reinstall in 4-way handshake

Practical impact

New KRACKs

Lessons learned
Idea 1: replay other handshake messages?
Idea 1: replay other handshake messages?

What if we replay Msg4?
MediaTek drivers vulnerable!

- Certain MediaTek Drivers accept replayed Msg4’s
- Used in 100+ devices → many vulnerable products

ASUS RT-AC51U

TP-Link RE370K
Idea 2: A/SNonce renewed during rekey?

AP can start new handshake to refresh the PTK
- Same messages exchanged as initial handshake
- New ANonce and SNonce must be used

macOS:
- Patched default KRACK attack
- But **reuses the SNonce during a rekey**
- SNonce reuse patched in macOS 10.13.3
Exploiting SNonce reuse

No problem if ANonce does change
› But Linux’s hostapd reused ANonce …
› Previous key was renegotiated and reinstalled
› Can decrypt old captured traffic!

Adversary can replay old handshake
› Tricky because messages must now be encrypted
› But feasible under specific circumstances
Idea 3: further audit patches

Several users reported: “Patched client still vulnerable to group key reinstallations”

› Either our patches are flawed …
› … or device always accepts replayed broadcast frames?!
No broadcast replay checks!

- Netis WF-2120
- AWUS036NH
- Nexus 5X

- 8 of out 16 tested devices vulnerable
- Likely caused by faulty hardware/firmware decryption
Related issue: group key improperly installed
Related issue: group key improperly installed

Contains key & current replay counter
Related issue: group key improperly installed

Contains key & current replay counter

Some install key using zero replay counter
Related issue: group key improperly installed

Affected devices:

› Samsung S3 LTE
› $POPULAR_CLIENT

How to abuse this?
GTK Install Attack

4-way handshake

Install GTK
GTK Install Attack

4-way handshake

Install GTK

BroadcastData(r)

Deauthenticate

Install GTK
GTK Install Attack

4-way handshake

Install GTK

BroadcastData(r)

Deauthenticate

Install GTK

Replay counter is reset to zero
GTK Install Attack

4-way handshake

Install GTK

BroadcastData(r)

Deauthenticate

4-way handshake

Install GTK

BroadcastData(r)
Idea 4: Impact of replaying broadcast frames?

Kankun smart power plug
› Android app to control it

Commands are broadcast UDP
› Destination MAC in payload (?!)
› Challenge/response protocol
Command Replay

CommandRequest

Challenge(id)

ConfirmRequest(id)

Ack

CommandRequest

Challenge(id)

ConfirmRequest(id)

Ack

New id

Run command
Command Replay

![Diagram showing the sequence of ConfirmRequest(id) and Ack for command replay.]

- ConfirmRequest(id)
- Ack
- ConfirmRequest(id)
- Run command
- Ack
Command Replay

ConfirmRequest(id) → ConfirmRequest(id)

Run command

Ack

Ack

ConfirmRequest(id) → ConfirmRequest(id)

Run command

Ack

Ack
Command Replay

Command again executed:
E.g. switch on/off
Is your device affected?

github.com/vanhoefm/krackattacks-scripts

› Tests clients and APs
› Works on Kali Linux

Remember to:
› Disable hardware encryption
› **Use a proper Wi-Fi dongle!**
Overview

Key reinstalls in 4-way handshake

Practical impact

New KRACKs

Lessons learned
Limitations of formal proofs

- 4-way handshake proven secure
- Encryption protocol proven secure

The combination was not proven secure!
Multi-party vulnerability coordination

Widespread issue! How to disclose?

Guidelines and Practices for Multi-Party Vulnerability Coordination (Draft)

Remember:
› Goal is to protect users
› There are various opinions
Conclusion

- Flaw is in WPA2 standard
- Proven correct but is insecure!
- Attack has practical impact
- Update all clients & check APs
Thank you!

Questions?

krackattacks.com
References


7. Multi-party vuln coordination

