密钥重载攻击：强制 WPA2 重用 Nonce

安全报告：密钥重载攻击：强制 WPA2 重用 Nonce
报告编号：B6-2017-101707
报告来源：360CERT
报告译者：360CERT
原文作者：Mathy Vanhoef, Frank Piessens
原文链接：https://papers.mathyvanhoef.com/ccs2017.pdf
保密级别：公开
更新日期：2017 年 10 月 17 日

目录

摘要

本文我们将介绍密钥重载攻击，这种攻击滥用了加密协议的设计和实现来重装一个已经在使用的 key。重置和 key 相关的参数，例如传输的 nonces 和重传计数器。一些加密的 Wi-Fi 握手包也会受到该攻击影响。

所有受保护的 Wi-Fi 网络使用四次握手来产生一个新的 session key。目前为止，握手协议已经使用了 14 年，一度被认为很安全，可以免受攻击。但是，本文证明了在密钥重载攻击面前四次握手协议是脆弱的。攻击者通过操作和应答握手信息来欺骗受害者重新安装一个已经在使用的 key。当密钥重载时，相关的参数例如传输包的增长序列（nonce）和重传计数器被重置为初始值。密钥重载攻击也能破坏 PeerKey，组密钥和 Fast BSS Transition（FT）握手协议。该影响取决于握手协议是否被攻击以及使用的数据加密协议。简单来说，针对 AES-CCMP 的攻击者可以应答和解密（不是伪造）握手包。这使得劫持 TCP 流并注入恶意数据变得有可能。而对于 WPA-TKIP 和 GCMP 允许应答、解密和伪造包影响是灾难性的。因为 GCMP 在传输的双方使用相同的 key，影响尤为严重。

最后，我们在实践中验证了我们的观点，并发现每个 Wi-Fi 设备都有可能受到一些变种攻击的影响。值得注意的是，Android 6.0 强制客户端使用初始值全为零的加密 key，使得我们的攻击对其影响是巨大的。

关键词： 安全协议；网络安全；攻击；密钥重载；WAP2；nonce 重用；握手包；包序列；初始向量

1 介绍

所有使用 WPA/2 协议的 Wi-Fi 网络并认为是安全的。更有甚者，由于 Hotspot2.0，现在公共的热点也能用这种加密认证[7]。所有这些技术都依赖于四次握手协议，该协议被定义在 802.11 中的 802.11i 修正案。本文我们将展示四次握手的设计流程和相关的握手包，因为我们认为这些握手包以及 WPA 和 WPA2-认证产品都会受我们攻击的影响。

四次握手提供了相互认证和会话密钥协商协议，连同（AES）-CCMP，数据保密和完整性协议，构成了 802.11i 修正案的基础。作为 802.11i 的核心部分，自从 2003 年第一次被命名为 WPA 被引入，它就免受了攻击。实际上，802.11i 目前已知的弱点是在（WPA-）TKIP[57,66]。其数据保密协议被设计作为破旧的 WEP 协议的一个短期解决方案。换句话说，TKIP 从来没有打算作为一个长期的安全解决方案。此外，虽然过去几年有几次针对 Wi-Fi 网络的攻击被发现，但都不是利用 802.11i 协议。而是利用 WPS[73]，有缺陷的驱动程序[13,20]，有缺陷的随机数字生成器[72]，可预测的预共享密钥[45]，不安全的企业认证[21]等等。在 CCMP 和四次握手里没有发现重大的弱点。这并不奇怪，毕竟，两者都被正式证明是安全的[39,42]。考虑以上几点，人们可以合理地假设四次握手确实是安全的。

尽管历史证明它是安全的，但我们可以证明四次握手面对密钥重载攻击是脆弱的。此外，我们发现了其他 Wi-Fi 握手协议中有相似的缺陷。这意味着，我们也能攻击 PeerKey 握手，组密钥握手和 Fast BSS Transition (FT) 握手。

我们攻击背后的原理在事后看来其实很简单。概括如下：

当客户端加入一个网络时，它会执行四次握手来协商得到一个新的会话。在接收到握手

的 Message 3 后，它会安装该 key。一旦 key 被安装，它将被用于数据保密协议来加密正常的数据帧。但是因为消息可能丢失，如果没有收到合适的响应作为应答，接入点（AP）将重发 Message 3。所以，客户端可能收到多次的 Message 3。每次接收到该消息，客户端就会重新安装相同的会话密钥，从而重置不断增长的传输包的序列号（nonce）以及数据保密协议使用的重传计数器。我们证明了攻击者可以通过收集和重发 Message 3 来强制重置 nonce。通过临时重置 nonce，可以攻击数据保密协议。例如，数据包可以被重放，解密，或伪造，这样的技术也可以用来攻击组密钥，PeerKey，fast BSS Transition。

当四次握手或 fast BSS 握手被攻击时，影响的大小取决于使用的数据保密协议。如果是使用 CCMP，任意数据包都可以被解密，反之，这可以用来加密 TCP SYN 包来挟持 TCP 连接。例如，攻击者可以注入恶意内容到未加密的 HTTP 连接中。如果是使用 TKIP 或 GCMP，攻击者可以解密并注入任意包。尽管 GCMP 是一个相对较新的 Wi-Fi 协议，预计在今后几年中会有很高的采用率。最后，当组密钥握手被攻击时，攻击者可以重放组寻址帧，即广播和多播帧。

我们的攻击对于 wpa_supplicant（Linux 系统常用的 Wi-Fi 客户端）2.4 版本和 2.5 版本影响巨大，该客户端将安装一个全零的加密 key 而不是重装一个真实的 key。该漏洞由 802.11 标准中的一句话引发：如果会话密钥已经被安装，建议从内存中清除部分的会话密钥[1, §12.7.6.6]。因为 Android 使用了改进的 wpa_supplicant，所以 Android 6.0 和 Android Wear 2.0 也包含此漏洞。因此，目前 31.2%的 Android 设备很容易受到这种攻击变种的破坏。[33]

有趣的是，我们的攻击并不违反四次握手和组密钥握手被证明的安全属性。特别值得一提的是，这些证明表明协商的会话密钥仍然是私有的，并且客户端和接入点（AP）的身份是被确认的。我们的攻击不会泄露会话密钥。（另外，虽然使用 TKIP 或 GCMP 可以导致正常的数据帧被伪造，但是攻击者不能伪造 EAPOL 消息，因此不能在握手期间伪装成客户端或 AP，所以不能模拟 key 的安装。不同的是，当一个协商 key 应该被安装时，他们的模型不能阐述。）实际中，这意味着相同的 key 可以被安装多次，从而重置 nonces 和数据保密协议使用的重传计数器。

总之，本文的主要阐述如下：

（1） 我们介绍了密钥重载攻击，攻击者强制重载一个已经使用过的 key，从而重置任何相关的 nonces 或重传计数器。

（2） 我们认为四次握手，PeerKey 握手，组密钥握手和 fast BSS transition 握手容易受到密钥重载攻击。

（3） 我们在实践中实施了我们的攻击，这表明所有的实现都易受某些变种的攻击。

（4） 我们评估了所有 802.11 的数据保密协议重用 nonce 的影响。

本文的其余部分如下：

第二节将介绍 802.11 标准的相关方面。第三节主要阐述针对四次握手和 PeerKey 握手的密钥重载攻击。第四节则是针对组密钥的描述。第五节描述针对 fast BSS transition 握手的攻击。在第六节，我们将描述攻击的影响，提出对策，解释安全性失败的原因，并讨论吸取的教训。最后我们会在第七节介绍本文的相关工作，并在第八节进行总结。

2 背景

这个章节主要介绍 802.11i 修正协议，在连入 WiFi 网络时的各种消息和协议和 802.11 中使用的各种数据机密性完整性协议。

2.1 802.11i 修正协议

在 WEP 安全性从根本上打破之后，IEEE 提供了一个更加健壮的解决方式，使用 802.11i 作为 802.11 的修正协议。修正协议定义了四次握手和两个数据机密性、完整性协议 WPA-TKIP 和 AES-CCMP。尽管 802.11i 修正案还在开发中，WiFi 联盟开始根据 802.11i D3.0 版本的草案来验证设备。验证程序叫做 WiFi Protected Access（WPA）。一旦 802.11i 的 D9.0 最终版本被批准，WPA2 的认证就建立在这个正式批准的版本上。因为 WPA 和 WPA2 都是基于 802.11，所以在技术水平上两者几乎相同的。最主要的区别是 WPA2 要求支持更安全的 CCMP，可选使用 TKIP，而 WPA 是相反的。

因为功能需求，WPA 和 WPA2 都采用四次握手保护 WiFi 网络。企业版网络也是依赖于四次握手。因此所有使用四次握手的 WiFi 网络都被这次攻击影响。

四次握手，组密钥握手和 CCMP 协议都正式被分析和证明是安全的

2.2 身份验证和连接

当客户端要连接 WiFi 网络，自动开始（互相）身份验证和连接。图 2 描述了连接阶段的握手。但是当第一次连接到网络时，是没有实际的身份验证。相反，使用了开放系统身份验证，对客户端进行身份验证。实际身份验证在四次握手中使用。但真正的身份认证仅在两个采用 fast BSS transition 握手协议的相同网络 AP 之间漫游时使用。

在开放式身份验证之后，客户端连接到网络中。通过客户端向 AP 发送一个连接请求完成。这条消息包含客户端希望使用的成对的密码组。AP 回复一个连接响应，通知客户端连接是否被成功建立。

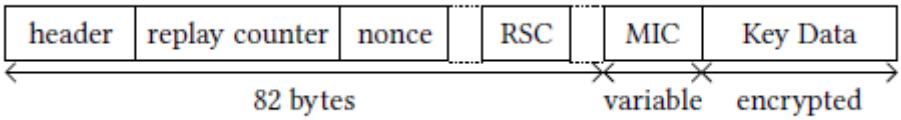

图 1

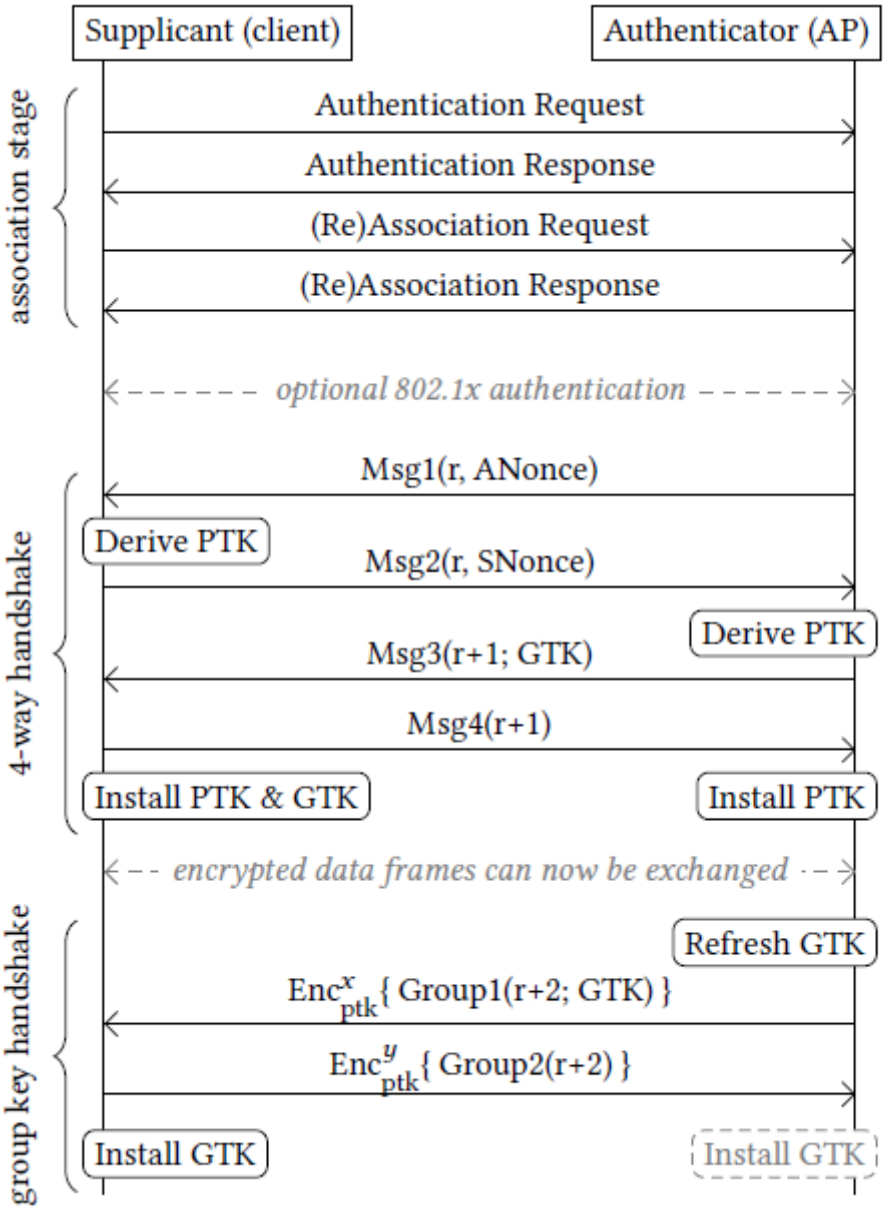

图 2

2.3 四次握手

四次握手提供相互身份验证，基于共享密钥技术，这种技术称为成对的主密钥 Pairwise Master Key(PMK)，并协商一个新的会话秘钥 PairWise Transient Key（PTK）。在这次握手中，客户端称为 supplicant，而 AP 称为 authenticator，PMK 由个人网络中的预共享密码生成，在企业网络中使用 802.1x 身份验证来进行协商。PTK 由 PMK，Authenticator Nonce (ANonce), Supplicant Nonce (SNonce)和 supplicant 和 authenticator 使用的 MAC 地址派生而来。一旦生成，PTK 被分割成确认 key(KCK)，加密 Key(KEK)，和临时 Key(TK)，确认 Key 和加密 Key 使用来保护握手消息，TK 和数据机密性协议是用来保护正常数据帧，如果使用了 WPA2，四次握手协议也传输现在的 Group Temporal Key 组临时密钥（GTK）到 supplicant。

四次握手中的每一条消息都是使用 EAPOL 帧格式。（如图 1）对字段进行介绍

首先，消息的头部定义了所代表的消息类型，我们将使用 message n 和 MsgN 来代表四次握手中第 n 段消息。

Replay count（重放计数器）字段用于检测重放的数据帧：authenticator 在发送一个帧之后会自增长，当 supplicant 对 authenticator 发送的 EAPOL 帧做出应答时，它使用相同的 replay count。

Nonce 字段是一个随机的 nones 值，这个随机值是 supplicant 和 authenticator 在生成新的会话秘钥这一步骤产生的。

接下来，如果 EAPOL 帧传输一个组密钥，RSC（接受序列）包含了 key 起始包号。

组密钥是存储在 Key Data 字段，使用加密 Key(KEK)加密。

最后，使用消息确认 Key(KCK)来进行完整性校验. MIC（Message Integrity Check）

图 2 表示了四次握手时消息的传输格式。我们使用一下的符号标记

MsgN(r, Nonce; GTK)

表示：四次握手中的第 N 条消息，重放计数器 replay count 为 r， 给定的 nonce,在';'之后的参数都存储在数据域中，也就是说会使用 KEK 加密。

1. Authenticator 通过发送 message 1 来初始化四次握手。包含 ANonce，是唯一一个没有 MIC(完整性校验) EAPOL 格式消息。
2. 当收到消息时，suplicant 生成一个 SNonce 而且导出 PTK，suplicant 发送 message2 给 authenticator，message2 包含了（SNonce）。
3. authenticator 收到 SNonce，也会导出 PTK。并且发送组密钥 GTK 给 supplicant。
4. supplicant 在安装 PTK 和 GTK 之后回复 message4，

authenticator 收到 message4 之后也会安装 PTK，其中 GTK 在 AP 启动时就已经安装。

1,2 条消息使用来传输 nonces，最后两条消息是用来传输组密钥 而且使用抵御降级攻击

注意：在已经存在的连接中，PTK 可以被刷新通过初始化新的四次握手。在密钥重载的过程中，所有的四次握手消息都是使用 PTK 完成数据机密性加密的。

2.4 机密性和完整性协议

802.11i 修正协议定义了两个数据级机密性协议，第一个是 TKIP（Temporal Key Integrity Protocol）暂时完整性协议。现在因为安全考虑 TKIP 被弃用。第二个是（AES-）CCMP，CCMP 是目前最广泛使用来保证数据机密性的协议，在 2012 年，802.11 修正协议中增加了新的数据机密性协议 Galios/Count Mode Protocol（GCMP）。这个修正协议也增加了在 60GHz 带宽的 short-range 信息交互，这需要一个能够快速计算的密码（fast cipher），比如 GCM。

现在 802.11ad 修正协议在 Wireless Gigabit（WiGig）中推广，而且预期在接下来几年被更快速广泛的采用。最后，802.11ac 修正协议通过加入 256 位的 key 更佳的拓展了 GCMP

当 TKIP 使用的时候，PTK 的一部分：TK（暂时密钥）被分隔称为一个 128 位加密密钥，两个 64 位的 MIC 消息完整性检测 key。第一个 MIC 使用在无线网热点对终端方向的消息发送，第二个则相反。这个加密使用了 RC4，每个包密钥都是独特的，通过 128 位加密秘钥，发送者的 MAC 地址，和增长的 48 位 nonce。Nonce 每发送一个帧都会自动增长，用做重放计数器，当安装 TK 时会重置为 1。通过 Michael 算法来确认消息的权威性，但是 Michael

算法是可逆的，给一个明文和 MIC 值，就可以有效的恢复 MIC 密钥。

CCMP 协议是基于 AES 在 CCM 模式下的加密方式（CBC-MAC 的计数器模式）。这是一种身份验证加密，使用了 Associated Data（AEAD）算法，只要特定的 Key 中没有重复的初始化向量就是安全的。在 CCMP 中，这个初始化向量是发送者 MAC 地址，48 位 nonce 和一些传输帧中额外 flags 的组合。这个 nonce 也会被用作接受者的重传计数器，每次发送后都会增加 1，再 TK 重新安装则会被初始化为 0.这样可以保证初始化向量不会重复。另外的，这个构造允许 TK 可以被直接做为信道两个方向交互的 key。

GCMP 协议是基于 AES-GCM，意味着使用了同样的加密计数器模式，得到的密文使用 GHASH 功能进行验证。和 CCMP 相同，是一个 AEAD 加密，只要在特定的 key 中初始化向量没有重复就可以保证安全，在 GCMP 中 nonce 作用也一样，在安装 TK 之后会被置为 0.通常保证初始化向量只使用一次，TK 可以作为交互双方的密钥，两个消息传递方向使用 TK 作为 key。一旦 nonce 被重复了，就有可能重构 GHASH 功能使用的验证密钥。

为了表示数据帧是通过数据机密性协议加密而且验证的，使用下面的符号标记

$$\mathrm{Enc}_k^n\{\cdot\}$$

n 表示nonce（也就是重放计数器）。参数k表示key，表示单播流量中的PTK。为了向组地址发送流量：比如广播和组播帧，使用GTK（组密钥）。最后使用两个标记符

Data(payload) 表示向单一地址发送

GroupData(payload) 向多个地址（组地址发送）

2.5 组密钥握手

Authenticator 周期性刷新组密钥，而且向所有的客户端发送组密钥，使用这些组密钥来进行握手。这些握手环节被证明是安全的，在图2的最后阶段。Authenticator 初始化所有的握手通过发送组消息 message1 给所有的客户端。Supplicant通过回复组message 2来确认收了了新的组密钥。取决于实现，Authenticator安装GTK可以在发送group Message1之后也可以在收到所有客户端的连接请求之后。最后group message 1 也包含了现有组密钥的收到重放计数器，这个字段是图1的RSC字段

所有的组密钥握手消息都是使用EAPOL格式帧，使用Group1和Group2在图2中代表。注意组消息1存储了新的组密钥在数据字段中（使用KEK确认密钥加密）、所以只要PTK被安装之后，完整的EAPOL格式帧都被被数据机密性协议保护

最后，当客户端发送一个广播或者多播帧，她首先需要发送单播给AP，AP对这个数据帧使用组密钥加密，然后广播给所有的客户端。这保证了所有的在AP范围内的客户端都能收到消息。

3 攻击四次握手

3.1 请求状态机

802.11i 的修订没有一个正式的描述 Supplicant 如何实现四次握手。相反，它只提供了如何实现的伪代码，并没有说明握手包将在何时被处理。幸运的是，802.11r 略微扩展了四次握手，并详细描述了请求状态机该如何实现。图 2 包括了这个状态机的简单描述。

当第一次连接一个网络并且开始四次握手的时候，状态机会进入 PTK_INIT 状态。在这里，它将初始化 PMK。当 Supplicant 接收到信息 1 的时候，会进入 PTK-START 阶段。这个经常在第一次连入一个网络的时候，或者是在前一个四次握手完成之后，会话密钥需要更新的时候。当进入 PTK-START 阶段，Supplicant 会随机生成一个 SNonce，计算一个临时 PTK（TPTK），并且在信息 2 中将 SNonce 发送给 Authenticator。Authenticator 随后会回复 Message 3，其将在 MIC 和重传计数器有效的时候被 Supplicant 所接受。如果成功，将进入 PTK-NEGOTIATING 状态，在这里若是 TPTK 和 PTK 相同时，Supplicant 会发送 Message 4 给 Authenticator。紧接着，就会进入 PTK-DONE 阶段，在这里 PTK 和 GTK 都会被装载，并且被用在数据保密协议以及密钥管理请求的完整性。最终，它会打开 802.1x 的端口，让 Supplicant 能够正常接收和发送数据包。注意，这个状态机会在 Authenticator 没接收到信息 2 和 4 的时候，分别重复发送信息 1 和 3。

我们确认 802.11r 的状态机符合在 802.11i 的修订中通过文字描述的状态机。最重要的是，我们能够进行重装密钥攻击的两个要素。第一，802.11i 的状态机中，AP 会在没有接受到回复的时候重传消息 1 和 3。因此，客户端必须接受这重传的信息 1 和 3，来和 802.11r 的状态机相匹配。更多的，802.11i 状态机要求客户端在接受并且回复 Message 3 之后要装载 PTK。这点也和 802.11r 中提供的状态机相匹配。

3.2 密钥重载攻击

我们的密钥重载攻击现在很容易理解：因为 Supplicant 即使在 PTK-Done 阶段，依旧持续接受重传的 Message 3，我们就可以强制重载 PTK。更确切地说，我们第一步在 Supplicant 和 Authenticator 中建立一个中间人（MitM）的身份。我们是用这个中间人的身份来进行 Message 3 的重放，并阻止 Authenticator 接受 Message 4。这就导致了，当重放 Message 3 的时候，Supplicant 就会重载一个已经被使用过的 PTK。反过来说，这个会重置被用在数据保密协议中的 Nonce。取决于不同协议的使用，这有可能造成数据包的重放、解密以及伪造。在 6.1 节中，我们会详细描述 Nonce 的重用会在不同协议中有什么实际的影响。

在攻击实践中，可能会遇到一些问题。首先，不是所有 Wi-Fi 设备都正确实现了状态机。特别是 Windows 和 iOS，它们不接受 Message 3 的重传。这和 802.11 标准是相违背的。总的来说，这些错误的实现并不会被我们的针对四次握手的密钥重载攻击所影响。不幸的是，从防御者的角度来说，iOS 和 Windows 设备依旧会被我们针对组密钥的攻击所影响。另外，因为他们都支持 802.11r，因此他们依旧有可能被针对 AP 的密钥重载攻击所影响。

第二个主要障碍是我们需要在 AP 和 Client 中获得一个中间人（MitM）的身份。这可能不能通过使用一个不同 MAC 地址的恶意 AP 在真实 AP 和 Client 间转发数据包来实现。在 2.3 节中提到，会话密钥（session key）是基于 Client 和 AP 的 MAC 地址来生成的，这意味着不

同的 MAC 会生成不同的密钥。这会导致握手过程以及攻击过程的失败。为了解决这个问题，我们可以部署基于频段的中间人攻击，在不同频段部署和目标 AP 相同 MAC 的恶意 AP 来实现。这个保证了 Client 和 AP 能产生一样的会话密钥。

第三个障碍是某些实现在 PTK 已经被加载的时候，只接受被数据保密协议所保护的数据包。这对我们的攻击是一个问题，因为 Authenticator 会在不加密的情况下重传 Message 3。这意味着这些重传信息会被 Supplicant 所忽略。即使这看起来会阻碍我们的攻击，但是我们发现了一个技术手段来绕过这个问题（3.4 节）

在接下来的两节中，我们会描述如何使用我们的密钥重载攻击在不同情况下来攻击四次握手实现的细节。更确切地说，我们会先描述在 Client（受害者）接受明文重传 Message 3 的情况。然后我们会演示对只接受加密重传 Message 3 的客户端的攻击。表 1 第 6 列里总结了哪些设备会在针对四次握手的不同的密钥重载攻击中受影响。不同设备的表现取决于操作系统以及被使用的无线网络设备。例如，即使 Linux 接受明文的重传 Message 3，但是在某些 Android 设备里使用的网卡会拒绝它们。因此，使用不同无线芯片的 Android 设备事实上可能会接受明文的重传 Message 3。

Table 1: Behaviour of clients: 2nd column shows whether retransmission of message 3 are accepted, 3rd whether plaintext EAPOL messages are accepted if a PTK is configured, 4th whether it accepts plaintext EAPOL messages if sent immediately after the first message 3, and 5th whether it is affected by the attack of Section 3.4. The last two columns denote if the client is vulnerable to a key reinstallation attack against the 4-way or group key handshake, respectively.

Implementation	Re. Msg3	Pt. EAPOL	Quick Pt.	Quick Ct.	4-way	Group
OS X 10.9.5	✓	✗	✗	✓	✓	✓
macOS Sierra 10.12	✓	✗	✗	✓	✓	✓
iOS 10.3.1 [c]	✗	N/A	N/A	N/A	✗	✓
wpa_supplicant v2.3	✓	✓	✓	✓	✓	✓
wpa_supplicant v2.4-5	✓	✓	✓	✓[a]	✓[a]	✓
wpa_supplicant v2.6	✓	✓	✓	✓[b]	✓[b]	✓
Android 6.0.1	✓	✗	✓	✓[a]	✓[a]	✓
OpenBSD 6.1 (rum)	✓	✗	✗	✗	✗	✓
OpenBSD 6.1 (iwn)	✓	✗	✗	✓	✓	✓
Windows 7 [c]	✗	N/A	N/A	N/A	✗	✓
Windows 10 [c]	✗	N/A	N/A	N/A	✗	✓
MediaTek	✓	✓	✓	✓	✓	✓

[a] Due to a bug, an all-zero TK will be installed, see Section 6.3.
[b] Only the group key is reinstalled in the 4-way handshake.
[c] Certain tests are irrelevant (not applicable) because the implementation does not accept retransmissions of message 3.

表 1

3.3 明文重传 Message 3

如果受害者在加载完会话密钥之后，依旧接受明文重传 Message 3 的话，我们的密钥重载攻击就很简单了。首先，攻击者会使用基于频段的中间人攻击，因此她可以控制握手包。然后她能够阻止 Authenticator 接受 Message 4。这就是图 4 里的第一阶段。在发送 Message 4 之后很快，受害者就会装载 PTK 以及 GTK 密钥。在这种情况下，受害者依然会打开 802.11x 端口，并且开始传输正常数据。注意一点，在数据保密协议中的第一个数据包使用的 Nonce 是 1。然后，在攻击的第三阶段，Authenticator 会因为没有收到 Message 4 而重传 Message 3。攻击者可以转发重传 Message 3 给受害者，导致它重新装载 PTK 和 GTK。总的来说，这样会重置数据保密协议使用的 Nonce 和防重传计数器。注意一点，攻击者不能够重放旧的 Message 3，因为 EAPOL 的重传计数器没有被刷新。我们现在暂时忽略攻击的阶段 4。最后，当受害者传输它下一个数据帧时，数据保密协议就会使用旧的 Nonce。这意味着攻击者可以在转发重传 Message 3 给受害者前等待任意的时间。因此，我们可以控制一定数量的会被重用的 Nonce。更多的，攻击者可以一直对 Client 进行掉线攻击（deauthenticating），直至它重新连入网络并执行新的四次握手的过程。

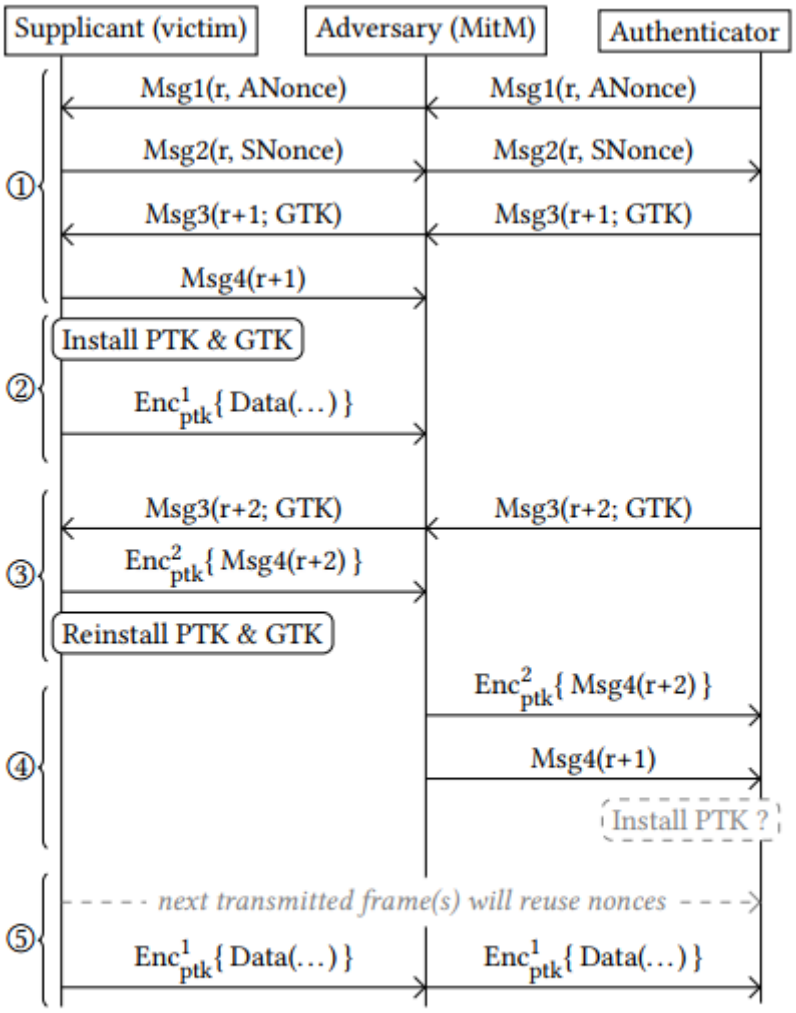

Figure 4: Key reinstallation attack against the 4-way handshake, when the supplicant (victim) still accepts plaintext retransmissions of message 3 if a PTK is installed.

图 4

图 4 同样展示了我们的密钥重载攻击会在 Message 4 由于背景噪声而丢失的时候自发地出现。不同的是，接受明文重传 Message 3 的 Client，可能会在没有攻击者的情况下重用 Nonce。在这种情况下，一个攻击者可以选择性地阻塞 Message 4，来区分攻击和随机背景噪声的干扰。

我们现在回到攻击的阶段 4。这个阶段的目标是完成 Authenticator 部分的握手阶段。这不是很重要，因为受害者已经加载了 PTK，这意味着它的下一个 Message 4 是被加密的。在 Authenticator 还未加载 PTK 时，它通常会拒绝已加密的 Message 4。因此，802.11 标准里揭示了 Authenticator 需要接受具有任意重传计数器的四次握手包，而不仅仅是最后一个。

接受 Message 4 的时候，Authenticator 验证 Key Replay Counter 字段值在当前四次握手过程中使用的。

在实践中，我们发现一些 AP 确实接受具有旧的重传计数器的数据包。更确切的说，一些 AP 接受在被发送给 Client 数据包中使用，并且还未被 Client 发送回来的重传计数器。这些 AP 会接受旧的未加密的 Message 4，就是图 4 里的重传计数器 r+1。总的来说，这些 AP

会装载 PTK，并且开始发送加密的单播数据帧给 Client。

虽然图 4 只说明了 Client 发送的 Nonce 重用情况，我们的攻击同样允许我们重放数据帧。首先，在 Client 在阶段 3 中重载 GTK 之后，AP 之后通过广播和多播的重传 Message 3 是可以被重放的。这是因为在重载密钥的时候，重传计数器也被重置了。第二，如果我们能够让 AP 装载 PTK，我们同样能重放从 AP 单播给 Client 的数据帧。

我们确认，在图 4 中的攻击在 MediaTek 的 Wi-Fi 设备和特定版本的 wpa_supplicant 下是有效的。我们将在下一节解释我们攻击的另外一种实现。

3.4 加密重传 Message 3

我们现在来解释我们是如何攻击那些一旦装载了 PTK 就只接受加密重传 Message 3 的 Client 的。为了完成这个，我们利用一个使用了数据保密协议的实体在执行四次握手的时候的一个内在的竞争条件。

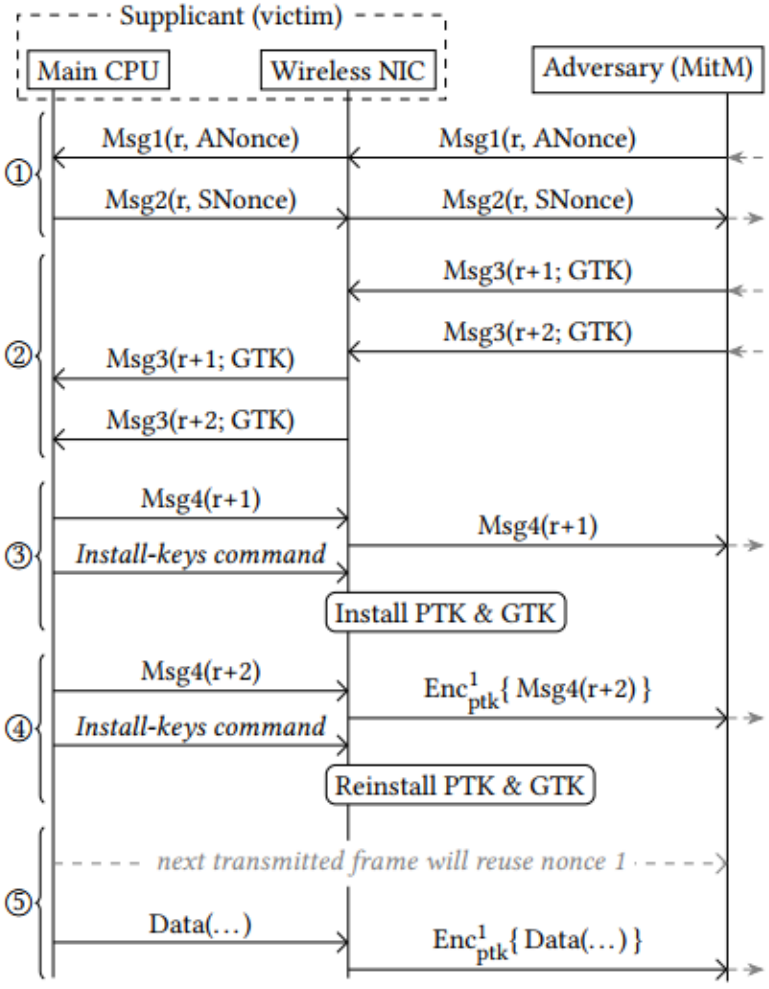

Figure 5: Key reinstallation attack against the 4-way handshake, when the victim accepts a plaintext message 3 retransmission if sent instantly after the first one. We assume encryption and decryption is offloaded to the wireless NIC.

图 5

就像先前说的，我们先来攻击 Android 上所实现的 Supplicant。在这里，我们发现当明文重传 Message 3 被紧接着原始 Message 3 发送的时候，Android 会接收这个重传信息。图 5 表明了为什么会发生这个情况，并且我们该如何利用它。注意一下就是，图 5 里并没有 AP，因为 AP 的反应在上文中已经描述的很清楚了。在我们的攻击中，首先让 Client 和 AP 交换信息 1,2。然后我们不将第一个 Message 3 转发给 Client，而是等着 AP 重传第二个 Message 3，在攻击的第二阶段，我们接连发送两个 Message 3 给 Client。这时候，实现了数据保密协议的无线网卡还没有装载 PTK，于是就将收到的两个包按顺序转发给 CPU。这个实现了四次握手的 CPU 接受了第一个 Message 3，并且让无线网卡装载 PTK。在攻击的阶段 4 里，Client 的 CPU 从接受序列里得到了第二个 Message 3。及时它注意到了这个数据包并没有加密，但是 Android 和 Linux 例外允许未加密的 EAPOL 包，因此 CPU 会执行这个重传 Message 3。因为网卡刚刚装载了 PTK，因此这个回复会是在 Nonce 为 1 的情况下加密的。在这之后，CPU 再次让无线网卡装载 PTK。同时，无线网卡也会重置与 PTK 相关的 Nonce 和重传计数器，这意味着下一个数据帧会重用 Nonce 1。

我们现在展示如何攻击 OpenBSD，OS X 和 macOS。这些设备只接受加密的重传 Message 3。就像攻击 Android 设备一样，我们在无线网卡和 CPU 间进行一个条件竞争。然而，我们现在的目标是四次握手重载密钥的过程。就像 2.3 解说的，所有的信息在重载密钥的时候都会经过数据保密协议的加密。

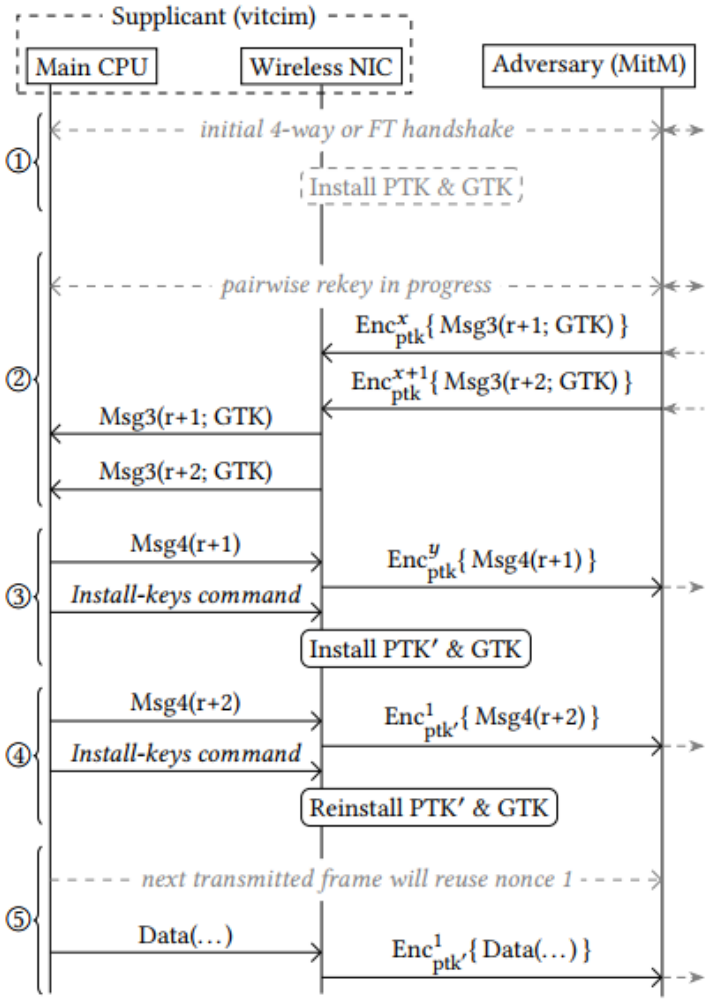

Figure 6: Key reinstallation attack against the 4-way handshake, when the victim only accepts encrypted message 3 retransmissions once a PTK is installed. We assume encryption and decryption is offloaded to the wireless NIC.

图 6

图 6 描述了这种攻击的细节。注意同样的 AP 并没有被展示在图里，因为从上文我们很清楚 AP 的反应会是怎么样。一样的，攻击者使用一个基于频段的中间人（MitM）身份。她让受害者和她自己进行一个初始的四次握手，并且在握手成功之后，初始化执行 PTK 的重载。及时她只能看到加密的数据帧，但是四次握手的信息能通过它们独特的长度和地址来探查。在这时候，攻击就像在 Android 那里一样，在攻击的第二阶段，攻击者没有立刻转发第一个 Message 3，而是等待 AP 重传 Message 3，然后将这两个 Message 3 按顺序一起发送给受害者。网卡会使用当前的 PTK 来解密这两个信息，然后将它们转发到 CPU 的接收序列。在第三阶段，CPU 会执行第一个 Message 3，让网卡装载新的 PTK。在第四阶段，CPU 会从接收序列取出第二个 Message 3。当 PTK 被装载的时候，OpenBSD，OS X 和 macOS（在这里被称为 CPU）会要求这个这个信息是加密的。然而，它们不会检查这个信息是使用哪个 PTK 密钥来加密的。因此，即使这个信息是通过旧的 PTK 来加密的，但是 CPU 依然会执行它。现在 Message 4 就会通过新的 PTK 和 Nonce 1 来加密并发送。在这之后，CPU 会让网卡重载 PTK，并且重置 Nonce 和重传计数器。最终，受害者发送的下一个数据帧就会使用新的 PTK 和 Nonce 1 来加密。我们确认这个攻击能在 OpenBSD 6.1，OS X10.9.5 和 macOS Sierra 10.12 里实现。

OpenBSD 仅在无线网卡卸载加密的时候受影响。比如，iwn 驱动和相关的设备支持硬件加密，因此他们就会收到攻击的影响。然而，rum 驱动在四次握手中使用的是软件加密，那么就不会被影响。

这种攻击技术需要我们等待会话密钥被重载的那一刻。一些 AP 会在每个小时都重载一次会话密钥。在实践中，Client 也会通过发送一个设置了 Request 和 Pairwise 位的 EAPOL 包给 AP 来请求重载会话密钥。很巧的是，Broadcom 的路由器并不会验证这些数据帧的真伪（MIC），这就意味着攻击者可以强制 Broadcom 的 AP 开始重载密钥的握手。讲这些结合到一起，我们可以让重载密钥发生，这就意味着攻击者可以实施重载密钥攻击了。

3.5 攻击 PeerKey 握手

PeerKey 握手和四次握手相关，被使用在当两个 Client 需要一个安全的方式来直接通信的时候。这由两个阶段组成。第一个阶段一个 STSL 和 SMK 的握手会被执行。它会在两个 Client 中协商密钥。而第二个阶段，一个新的会话密钥会通过主密钥产生，并且通过 STK 握手来传输。虽然这个协议没有被广泛的支持，但是它很好的展示了我们的密钥重载攻击良好适应性。

不出所料，SMK 握手并不会被我们的攻击所影响。在这之后，主密钥的协商握手并没有被数据保密协议所保护，这就意味着它们没有 Nonce 和重传计数器来重置。然而，STK 握手是基于四次握手的，它装载了用于数据保密协议的密钥。因此，他可以像四次握手那样被攻击。这种攻击在 wpa_supplicant 上通过了测试。为了实现这个测试，我们修改了一个 wpa_supplicant 实例，让它发送第二个（重传）Message 3。这个证实了未修改的 wpa_supplicant 会在 STK 握手中接收到重传 Message 3 时重载 STK 密钥。然而，我们没有发现其他支持 PeerKey 的设备。因此，我们针对 PeerKey 握手的攻击的影响范围很小。

4 攻破组密钥握手

在这一节，我们会在组密钥握手中，实现我们的重载密钥攻击。我们展示所有的 Wi-Fi 设备都会被这攻击所影响，让攻击者可以重放广播和组播的数据帧。

Table 2: Behaviour of Access Points. The 2nd column shows whether it accepts replay counters it used in a message to the client, but did not yet receive in a reply, or if it only accepts the latest used counter. Column 3 shows whether the GTK is installed immediately after sending group message 1, or if this is delayed until all clients replied with group message 2.

Implementation	Replay Check	GTK Install Time
802.11 standard	not yet received	delayed
Broadcom	latest only	immediate
Hostapd	not yet received	delayed
OpenBSD	latest only	delayed
MediaTek	latest only	immediate
Aironet (Cisco)	latest only	immediate
Aerohive	not yet received	delayed
Ubiquiti	not yet received	delayed
macOS Sierra 10.12	latest only [a]	immediate
Windows 7	latest only	*The group key is*
Windows 10	latest only	*never refreshed*

[a] Retransmitted handshake messages do not use a new replay counter, so at all times there is only one allowed value.

表 2

4.1 组密钥握手的细节

网络会定期更新组密钥来保证只有近期认证的 Client 拥有这个密钥。在多数防御情况下，组密钥会在 Client 离开网络的时候进行更新。新的组密钥会通过组密钥握手进行分发，而这个握手过程已经被正式地证明是安全的。就像图 2 中展示的那样，握手过程由 AP 发送信息 1 给所有的 Client 开始。AP 也会在没有收到回复的时候重传这个信息 1。注意一下，EAPOL 的重传计数器在这些重传信息中总是增加 1。在我们的攻击中，我们的目标是收集一个重传的组信息 1，阻止它到达 Client，然后在一段时间后再转发给 Client。这会让 Client 装载组密钥并重置重传计数器。

我们攻击的第一个条件是 Client 会在装载一个已经用过的组密钥的时候重置重传计数器。因为 Client 同样使用 MLME-SETKEYS 请求安装组密钥，这种情况就会发生。我们确定在实际中所有的 Wi-Fi 设备都会在装载一个用过的组密钥的时候重置重传计数器（表 1 列 7）。因此，所有的 Wi-Fi 设备都会被我们随后的攻击所影响。

第二个攻击条件是我们必须能够收集一个仍然能被 Client 所接受的组信息 1，然后这个信息中包含了已经被 AP 使用过的组密钥。如何去得到这个信息取决于 AP 何时开始使用新的组密钥。特别的事，AP 将会在发送第一个组信息 1 之后立即使用新的组密钥，或者会延迟装载知道所有的 Client 都回复了组信息 2 表明收到了组信息 1。表 2 第 3 列总结了一些 AP

的行为。注意根据标准，新的组密钥需要在收到所有 Client 的组信息 2 之后才能被装载。例如 GTK 会被延迟装载。当一个 AP 会立即装载新的组密钥的时候，我们的密钥重载攻击就很简单了。然而，如果 AP 延迟装载组密钥，我们的攻击就会更复杂。我们将会在 4.2 和 4.3 节中更详细地讨论这两种情况。

就像 2.3 节中所说，AP 会广播和组播通过组密钥加密的数据帧。因为我们的密钥重载攻击目标是 Client，这就意味着我们不能在加密的时候强制 Client 重用 Nonce。然而，Client 会在重载组密钥之后重置重传计数器，我们就可以重放这些数据帧给 Client。

大多数 AP 每小时都会更新一次组密钥。一些网络会在每次有设备离开网络的时候更新组密钥。而且，Client 可以通过发送一个设置了 Request 和 Group 位的 EAPOL 包来触发组密钥的握手。同样，Broadcom 路由器并不会验证这些信息是否正确，这就意味着攻击者可以伪造它们来触发组密钥的更新。将这些结合起来，我们就可以假设大多数网络会进行组密钥的更新，然后我们就可以实行接下来所说的攻击。

4.2 攻击快速装载密钥的 AP

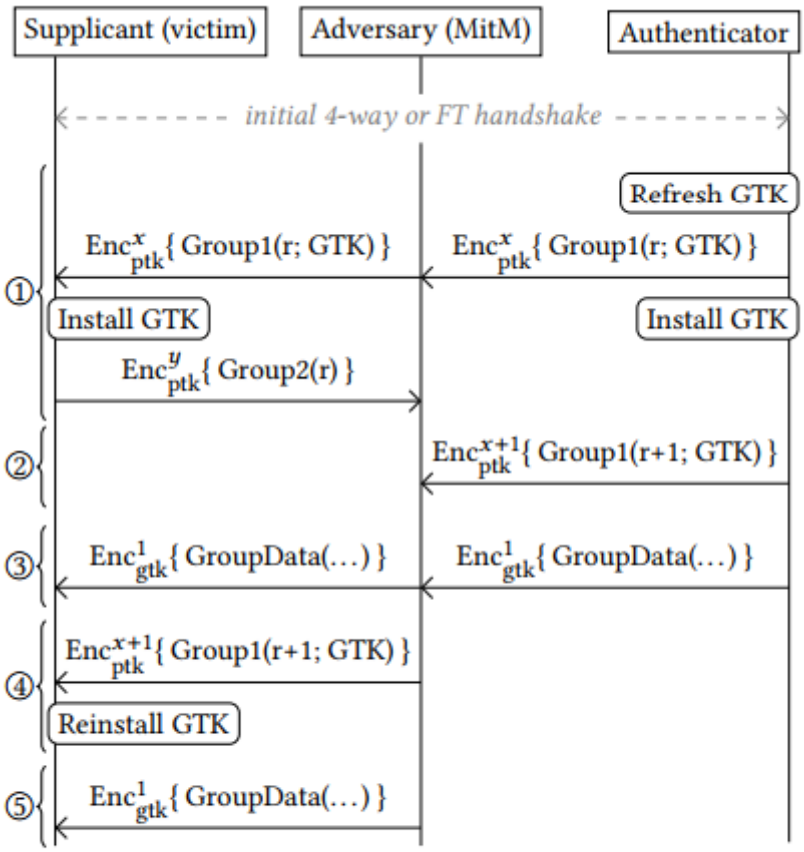

Figure 7: Key reinstallation attack against the group key handshake, when the authenticator (AP) immediately installs the GTK after sending a Group Message 1 to all clients.

图 7

图 7 展示了当一个 AP 会在发送完给所有的 Client 的组信息 1 之后立即装载组密钥的情

况下，我们的密钥重载攻击将会如何实现。注意组密钥的握手信息是通过当前 PTK 进行数据保密协议来加密的。作为组信息 1 的回复，Client 会安装新的 PTK 然后返回组信息 2。攻击者可以阻止 AP 接收这个组信息 2，然后 AP 就会在攻击的第二阶段重传一个新的组信息 1。我们现在等待一个广播数据帧的发送，然后将其转发给受害者。在这之后，我们转发阶段二中的重传组信息 1。这样就会导致受害者会重载 GTK，从而会重置相关的重传计数器。这就允许我们来重放广播的数据帧（阶段 5）。Client 会接收这个数据帧，因为它的重传计数器被重置了。

我们必须在一个广播数据帧之后再发送重传组信息 1。这是因为组信息 1 包括了当前使用的组密钥的值以及重传计数器。因此，如果在广播帧之后发送，它就会包含一个更新过的重传计数器，那么我们就不能使受害者的重传计数器重置。

我们确定这种针对快速装载组密钥的 AP 在实践中是有效的。基于我们的实验，所有的 Wi-Fi 设备在连接这类 AP 之后，都会受到影响。

4.3 攻击延迟装载密钥的 AP

通过组密钥握手来攻击延迟装载密钥的 AP 是比较复杂的。注意上一个攻击方式会因为图 7 中的第三阶段里那个广播数据包是由旧的组密钥加密而失败。因为在这种情况下，AP 没有接收到从所有 Client 返回的组信息 2，就不会装载新的组密钥，也就不能使旧的组密钥的重传计数器重置。

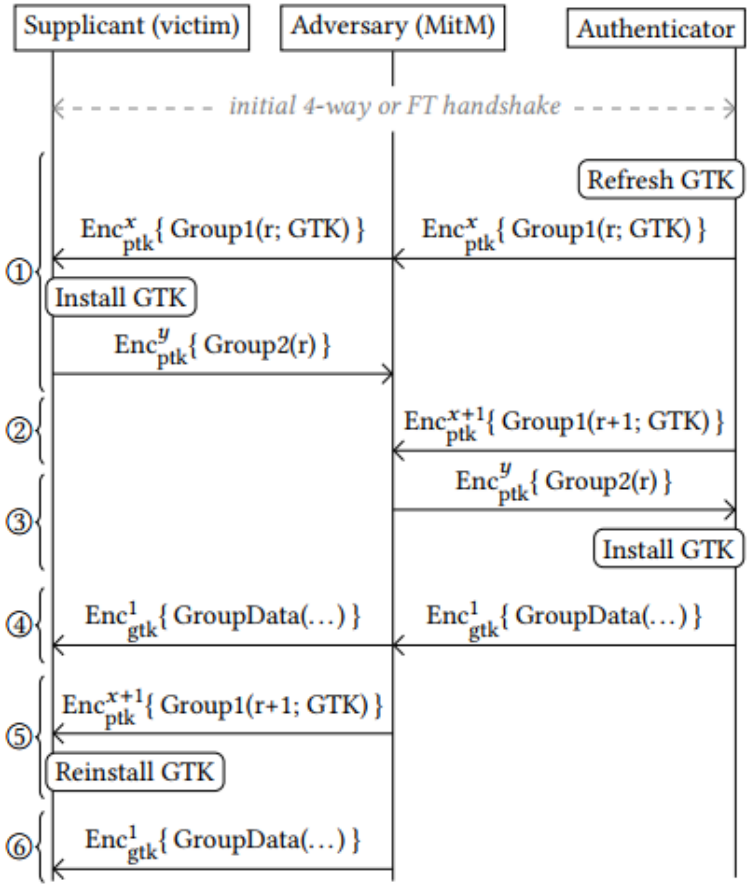

Figure 8: Key reinstallation against the group key handshake, when the AP installs the GTK after accepting replies with a non-yet-received replay counter from all clients.

图 8

图 8 展示了一个能够解决这个问题的方法。这种攻击的前两个阶段和上一个攻击类似。就是 AP 产生新的组密钥，然后将其传输给受害者，然后攻击者阻止 AP 接收 Client 发送的组信息 2。这会导致 AP 重传组信息 1，并且 EAPOL 的重传计数器为 r + 1。在攻击的阶段三，我们转发旧的组信息 2（重置计数器值为 r）给 AP。有趣的是，AP 应该接受这个信息，及时它没有使用最新的重置计数器的值（r + 1）：

在接收组信息 2 的时候，AP 需要验证重传计数器的值为曾经在组密钥握手过程中使用过的值。

这个标准不需要重传计数器的值和 AP 最新的值相匹配，而是需要和在组密钥握手过程用过的值中的一个相匹配即可，也就是任意的组信息 1 使用的重传计数器的值。在实践中我们发现，一些实现确实接受了这个旧的还未收到回复的重传计数器的值（表 2 列 2）。因此，这个 AP 就会装载新的组密钥。在这种情况下，攻击就会想之前那个一样进行。直到第一个广播数据包被传输，我们就可以在攻击阶段 5 里进行重载组密钥的操作，然后在第 6 阶段中重放广播数据帧。

同样的，我们必须在一个广播数据帧之后再发送重传组信息 1。不然我们的组信息 1 会包含一个更新过的重传计数器。

我们在一些延迟装载 GTK 的 AP 上进行了攻击测试，然后他们接受了先前在和 Client 发

包时的还未收到回复的重传计数器（表 2 列 2）。我们已经知道所有的 Wi-Fi 客户端会在重载 GTK 的时候重置重传计数器，因此他们都会被这种攻击所影响。最后，一个 OpenBSD 的 AP 并不会被影响，因为它是延迟装载 GTK 的，并且只接受最新的重传计数器而不是旧的。

5 针对 802.11 RFT 握手的攻击

在这一节中，我们将介绍 Fast BSS Transition（FT）握手并证明它的实现也受我们密钥重载攻击的影响。

5.1 Fast BSS Transition（FT）握手

802.11r 修正案增加了 Fast Basic Service Set（BSS）Transition（FT）握手到 802.11[5]。目的是减少客户端从一个 AP 到另一个相同网络的漫游时间。（例如相同的基本服务集）此外还需要一个握手，包括新的 802.1x 和四次握手（请看图 2）。然而，因为 FT 握手依赖于网络之前连接中派生的主密钥，所以并不需要一个新的 802.1x 握手。此外它将四次握手的阶段嵌在了认证和重组框架里。

一个标准的 FT 握手如图 9 的阶段 1 所示。

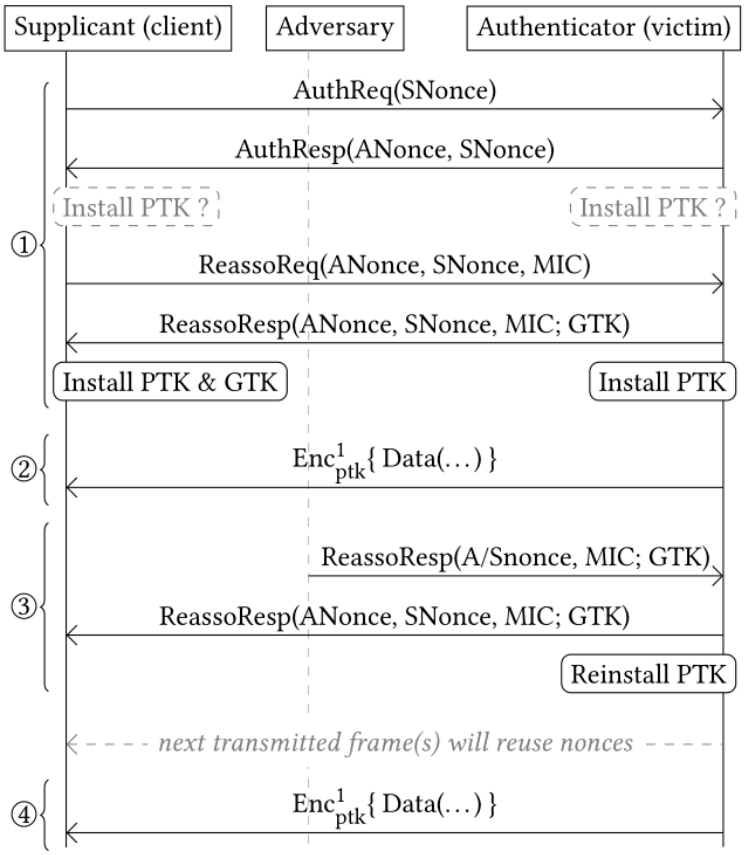

图 9 针对 Fast BSS Transiton（FT）的密钥重载攻击，注意到它并不需要中间人，仅仅是能窃听和重放帧

图中可以观察到 FT 握手不像四次握手，它由请求者初始化。前面两个消息是认证请求（AuthReq）和认证响应（AuthResp）。它们的功能相当于四次握手的 Message 1 和 2，产生随机数来作为 nonces（新的会话密钥）。在这之后，客户端发送一个重新连接请求（ReassoReq），AP 回复一个重新连接响应（ReassoResp）。它们和四次握手的 Message 3 和 4 相似，用于完成 FT 握手和传送 GTK 给客户端。

只有两个重新连接的消息是使用 MIC（见图 9）验证的，另外，FT 握手中没有包含重传计数器的消息。相反地，在不同握手调用之间，FT 握手依靠随机的 SNonce 和 ANonce 来提供重放保护[1, §13.5.2]。

根据标准，在认证响应被发送或接收后必须安装 PTK [1 , §13.9]。这在图 9 的第 1 阶段的灰色方框里已经标注了。另外，802.1x 的逻辑端口只有在发送或接收重连请求后才会被打开，这保证了即使握手阶段 PTK 已经被安装了，AP 和客户端也只有在握手完成后才能传输和接收数据帧。结合起来，表明了 802.11r 修正案里定义的 FT 握手并不容易受到密钥重载攻击。然而，通过实验和代码检查，我们发现大多数协议实现都是在发送或接收完重连响应后安装 PTK 和 GTK。这种行为已经在图 9 的阶段 1 的黑色方框里阐述了。结果，在实际中，FT 握手的实现容易受到密钥重载的攻击。

5.2 针对 AP 的密钥重载攻击

由于 AP 是在响应一个重连请求后才安装 PTK，所以我们的目标是重放该帧。我们注意到，在实际中，AP 必须接受重连请求的重传。这是因为 AP 的重连响应可能由于背景噪声而丢失，所以需要客户端重新发送一个请求。

图 9 显示针对 FT 握手的密钥重载攻击。我们并不需要中间人，只需能够窃听和注入帧就行。在攻击的第一阶段，我们让客户机和 AP 执行一个正常的 AP 握手，然后等待 AP 传输一个或多个加密数据帧，之后我们重放对 AP 的重连请求包，因为 FT 握手不包含重传计数器和有效的 MIC，所以 AP 将会接受并处理该重放帧。最后，AP 将会在攻击的阶段 3 重装 PTK，随之重置相关的 nonce 和重传计数器。因此，AP 发送的下一个数据帧将会用已经用过的 nonce 进行加密。和之前的密钥重载攻击类似，它可以让攻击者重放之前客户端发给 AP 的数据包。我们认为该攻击对 FT 握手的危害尤其大是因为 FT 的消息不包含重传计数器，这使得攻击者可以不停地重放重连请求，不停地重置 AP 使用的 nonce 和重传计数器。

我们测试了三个支持 802.11r 的 AP。第一个是用开源的 hostapd，第二个联发科设计用于家用路由器，运行在 Linksys RE7000 上的。第三个是一个专业的 Aerohive AP。以上三个都受密钥重载攻击的影响。

由于背景噪声，重连响应包如果丢失，客户端将会自动重发重连请求包，这会导致 AP 密钥重载。也就是说，就算没有受到攻击，AP 也已经重用 nonces 了。

FT 握手没有使用数据保密协议进行额外的保护。特别是管理帧保护（MFP），它不保护认证和重连帧[1, §12.6.19]。因此，FT 握手是否启用 MFP 对密钥重载攻击的防御是微不足道的。

5.3 滥用 BSS 传输请求

FT 握手只有从一个站点 AP 漫游到另一个站点 AP 时才会被实现。所以攻击发生的场景

是有限制的。然而我们可以强制受害者进行 FT 握手，具体方法如下：
首先假设一个客户端连接到了一个支持 802.11r 的网络，其次在客户端的网络里没有其他的 AP，我们就可以利用虫洞攻击[41]克隆一个真实的 AP 网络放在客户端附近。这使得客户端认为目标网络的另一个 AP 就在附近。最后我们向客户端发送一个 BSS Transiton Management 请求。该请求帧用于负载均衡[1, 11.24.7]，并命令客户端漫游到另一个 AP。BSS 是一个未经验证的管理框架，所以可以被攻击者伪造。最后客户端接受该请求帧，并使用 FT 握手漫游到伪造的 AP。

我们测试了支持 802.11r 的客户端，证明 wpa_supplicant，iOS[8]，Windows 10[52]都会接受伪造的传输请求，并用 FT 握手漫游到另一个 AP。

6 评估和讨论

在这一节中我们评估nonce重用对于802.11数据机密性协议的影响，展示攻击场景，和明确漏洞实现，解释为什么安全证明错过了我们的攻击，并提出了对策。

6.1 802.11 协议中重用 nonce 造成的影响

Nonce 重用造成的影响取决于使用的数据机密性协议。对于 TKIP，CCMP，GCMP。三个协议使用了流密钥来加密帧。因此，对 nonce 的重用总是意味着重用 keystream。这可以用于解密数据包。在我们的攻击中受害者的重放计数器会被重置，因此，这三种协议都容易受到重放攻击的攻击。

当使用 TKIP 时，我们也可以像下面一样恢复 MIC。

1.我们滥用 nonce 重用来解密一个完整的 TKIP 包，包括它的 MIC 字段。

2.攻击 Michael 算法，使用给出的明文帧和 MIC 密文，来恢复 MIC 明文。因为 TKIP 使用了不同的 MICkey 在每个不同的消息传输方向，我们可以再特定的方向上伪造数据帧。方向的源头是被重装密钥攻击的设备。表 3 在提到 TKIP 时总结了这一点

当使用 CCMP 时，实际的攻击被限制为对包的重放和解密。尽管有一些消息伪造攻击讨论，但这些攻击是理论上的，不能用于伪造消息。

当使用 GCMP 时，其影响是灾难性的。

1.可以对包进行重放和解密。

2.有可能恢复身份验证密钥，它在 GCMP 中用于交互双方两个方向上的通信，因此，与 TKIP 不同的是，攻击者可以在两个方向上伪造数据包，鉴于 GCMP 预计将在未来几年在以 WiGig 的名字广泛采用，这是一个令人担忧的情况。

一般来说，对手总是可以在特定的通信方向上重放、解密或伪造数据包。具体的方向取决于握手的方式。例如,通过四次握手攻击客户端，它可以用于

1.重放单播和广播/多播帧给客户端

2.将客户端发送给 AP 的帧的解密

3.伪造客户端发送给 AP 的帧

	Replay [c]	Decrypt [a]	Forge
4-way impact			
TKIP	AP → client	client→ AP	client→ AP [b]
CCMP	AP → client	client→ AP	
GCMP	AP → client	client→ AP	client↔ AP [b]
FT impact			
TKIP	client→ AP	AP → client	AP → client
CCMP	client→ AP	AP → client	
GCMP	client→ AP	AP → client	AP ↔ client [b]
Group impact			
any	AP → client [c]		

[a] With this ability, we can hijack TCP connections to/from an Internet endpoint and inject data into them.
[b] With this ability, we can use the AP as a gateway to inject packets towards *any* device connected to the network.
[c] This denotes in which direction we can replay unicast and group-addressed frames. For the group key handshake, only group-addressed frames can be replayed.

表 3：密钥重载攻击对使用了四次握手，FT，组密钥握手数据机密性的协议影响。每一个小格都展示了在这个方向上的数据帧可以被替换、解密或者伪造

然而，攻击 FT 握手行为中攻击了 AP 而不是客户端，意味着，攻击者可以在相反的方向重放，解密或者伪造数据包。表 3 在附录总结中，考虑了握手被攻击

最后，在不同的情况下，我们可以从客户机向 AP 发送消息(参见表 3)，有趣的是，AP 通常不是数据帧的最终目的地，而是将数据帧转发。这意味着我们可以伪造数据包发视频发送到任何连接到网络的设备上。根据 AP 的不同，甚至可以发送一个数据帧反射回到客户机。

6.2 攻击场景

在其他方面，密钥重载攻击可以让攻击者解密 TCP 数据包，知道传输序号，挟持 TCP 数据流注入任意数据。对 wifi 网络最常见的攻击之一是:在未加密的 HTTP 连接中注入恶意数据。

重放广播和多播帧的能力。比如：组帧，也是一个明显的安全违规。为了说明这将如何影响实际系统，请考虑在广播模式下运行的网络时间协议(NTP)。在这种模式下，客户端首先通过一个初始化过程，然后通过监听身份验证的广播 NTP 数据包来同步它的时钟。Malhotra 和 Goldberg 已经表明，如果这些广播包被重放，受害者将永久被困在一个特定的时刻。使用我们的组密钥攻击可以重放这些帧，即使它们是通过一个受保护的无线网络发送的。注：以这种方式操作时间会破坏安全性，例如 TLS 证书，DNSSEC 、Kerberos 身份验证和比特币。另外一个例子是 xAP 和 xPL 家庭验证协议。这些使用 UDP 广播包给设备发送命令。我们推测，密钥重载攻击允许我们重放这些命令。所有这些例子都说明，重播广播或多播帧的影响不应被低估。

6.3 全零加密密钥漏洞

密钥重载攻击对于四次握手的攻击发现了 wpa_supplicant 的特殊行为

1. 2.3 版本和更低的版本在很容易受到攻击，没有任何意外的副作用。
2. 我们发现当接收到重新传输的 message3 时，2.4 和 2.5 版本安装了一个全零加密密钥(TK)。这个漏洞似乎是由 802.11 标准中的一句话引起的，该标准间接建议在安装了 TK 之后，从内存清除它。
3. 版本 2.6 修复了这个 bug，只在第一次接收 Message 3 时安装了 TK。
   然而，在修补这个 bug 时，只考虑了一个良性的场景：message3 被重新传输是因为 message 4 由于背景噪音丢失。他们没有考虑到一个活跃的攻击者可以滥用这个漏洞来强制安装一个全为零的密钥。

   因此，补丁并没有被视为严重的安全问题，也没有被移植到较老的版本中。独立于这个 bug，所有版本的 wpa_supplicant 在接收时重传的 message3 时重新安装组密钥，也容易受到第四章所说的组密钥攻击。

   因为 Android 内部使用的是一个稍微修改过 wpa_supplicant 的版本，它也会受到这些攻击的影响。特别地，我们检查了 Android wpa_supplicant 关键源代码，并发现所有的 Android 6.0 版本都包含了全零加密密钥的漏洞。android wear 2.0 也很容易受到这次攻击。尽管第三方厂商可能会在他们的 Android 版本中使用不同的 wpa_supplicant 版本，但这说明大多数 Android 6.0 版本都容易受到攻击。换句话说，31.2%的 Android 智能手机很容易受到全零加密密钥漏洞的影响。最后，我们还从经验上证实，Chromium 很容易受到全零加密密钥漏洞的影响。

6.4 安全证明的限制

有趣的是，我们的攻击并没有违反对四次握手和组密钥握手的安全属性。

1. 他人证明了四次握手提供了密钥保密和会话认证。密钥保密表明只有 authenticator 和 supplicant 将会拥有 PTK。因为我们没有恢复 PTK，所以这个仍然正确。会话身份验证使用了匹配会话的标准概念。直观地说，协议时安全的除非攻击者能依赖消息参与到完成这个协议中。我们的攻击，包括我们使用的基于信道的中间人攻击，并不违反此属性：我们只能通过转发(转发)消息使端点完成握手。
2. 他人证实了组密钥握手中 key 的有序性和 key 保密性。密钥有序性确保 supplicant 不安

装旧的 GTK。这在我们的攻击中仍然是正确的，因为我们重新安装了新的的组密钥。另外，我们不知道组密钥，因此我们的攻击也不违反密钥保密。

然而，这些证明并没有对密钥安装进行建模。不同的是，它们不会声明密钥重载使用了数据机密性协议。在实践中，这意味着相同的密钥可以多次安装，重置为了保证数据机密性中使用的 nonce 和重放计数器

6.5 对策

密钥重载攻击可以在两层上减轻。

1. 实现数据保密协议的实体应该检查是否已经安装了已经使用的密钥。如果是这样，它不应该重置相关的 nonces 和重放计数器。这可以防止我们的攻击，至少攻击者不能临时欺骗去实现在重新安装当前的 key 之前安装一个不同的(旧的)密钥。特别是，当使用这个对策时，接收组密钥的握手重放计数器只会增加。否则，攻击者可以使用旧的 Group message1 来临时安装一个旧的(不同的)密钥，然后使用现有的 group message1 重新安装新的组密钥。
2. 第二种解决方案是确保在握手过程中，在实现数据保密协议的实体中只安装一个特定的密钥。例如，在四次握手过程中生成的会话密钥应该只安装一次。当客户端接收到一个重新传输的 message3 时，它应该回复，但不能重新安装会话密钥。这可以通过在图 3 的状态机中添加一个布尔变量来实现。它被初始化为 false，当在 PKT-START 时生成一个新的 PTK 时，它将被设置为 true。如果在输入 PTK-DONE 且布尔值为 true，则会安装 PTK 并将布尔值设置为 false。如果在输入 PTK-DONE 时布尔值为 false，则跳过 PTK 的安装。注：这正是 2.6 版本的 wpa_supplicant 实施的。

证明上述对策的正确性:我们 NuSMV 中对改进状态机进行了建模，并使用该模型证明了两个安装 key 总是被新的 PTK 分离。这意味着相同的密钥从未安装过两次。注：key 保密性和会话身份验证已经在其他工作中得到了验证。

我们目前正在通知供应商关于我们发现的漏洞，这样他们就可以实施这些措施。我们所熟知的各种攻击的变种将会被提供给一个供应商名单。

6.6 讨论

从我们的结果中可以学到一些重要的教训。首先，协议的规范应该是足够精确和明确的。例如，当我们在第 3.3 节中攻击四次握手时，我们注意到 802.11 标准对于应该被接收的重放计数器的值是不明确的。更精确或更正式的规范将避免这种潜在的不正确的解释。

其次，这并不是因为协议已经被正式证明是安全的，它的实现也是安全的。在我们的例子中，在正式证明中使用的四次握手模式并没有完全反映现实。这是因为它没有定义何时应该安装协商的会话密钥。因此，不能保证只安装一次会话密钥。只有当我们读代码才发现正式模型并没有和显示情况相匹配，这些 key 可以重新安装。在这方面，正式的证明实际上可能会适得其反:一旦一个协议被正式验证，社区可能会对审核实际的实现变得不那么感兴趣。

有趣的是，一个模型可能是错误的，因此并不能准确地反映现实，也同样适用于我们自己的对策的证明。换句话说，不是因为我们在 NuSMV 中建模了对策，所有的实现现在都突然变得安全了。在现实中，我们的正式状态机可能无法准确地反映某些实现，供应商实现可能有缺陷，或者供应商可能受到未知的攻击的影响。因此，保持审计和测试实际实现是非常

重要的。

另一个教训是，数据保密协议应该为 nonce 重用提供一些保护。例如，对于 GCMP，可以在 nonce 重新使用时恢复身份验证密钥，而对于 CCMP 来说则不是这样。更广泛地说，应该使用一种 nonce 抵制重用加密模式，比如 aes-siv、gcm-siv 或 hs1-siv 。这些减少了 nonce 重用的影响，也降低了 key 重新安装攻击的影响

7 相关工作

本章节我们将探索密钥重载攻击的历史，并对其它 Wi-Fi 和协议安全工作进行一个概述。

7.1 密钥重载攻击

我们并不清楚密钥重载攻击之前的一些研究，这很可能因为之前 Wi-Fi 握手包密码仍然易受到攻击。我们也是现在才发现长达 14 年的四次握手会受到密钥重载攻击的影响。此外，该缺陷不仅存在实现中，而是在协议规范（标准）本身。

电源故障也会导致 nonce 被重用。电源故障后，key 从 boot 的非易失性存储器里被恢复，但是 nonce 会被重置为初始值。对于这个问题，Zenner 在参考文献[76]提出了解决方案。然而，和密钥重载攻击不同的是，触发电源故障不能通过网络进行远程实施。这需要对被攻击的设备进行物理访问。另外，电源故障不影响我们所研究的协议的安全性，因为这些握手协议是用来避免维护新旧连接的状态。

在参考文献[16]，Bock 等人发现一些 TLS 服务器正在使用静态的 nonces，这是由 TLS record layer 协议的错误实现引起的。就是说，它不是由重装一个已经使用过的 key 引起的。此外，有些服务器使用随机产生的 nonces，这意味着在实践中很可能因为生日悖论攻击导致 nonce 重用。相反，密钥重载攻击允许攻击者通过重放握手信息强制重用 nonce，这是由于握手协议的规范（或实现）而导致的缺陷。

McGrew 写了一篇关于产生 IVs 和 nonces 的最好实现的调查报告，并总结了它们是如何在协议中被生成和使用的[51]。然而在讨论安全风险时，没有提到密钥重载攻击。

另一个相关的工作是 Beurdouche 等人[14]和 de Ruiter， Poll[27]所进行的。他们发现了几个 TLS 实现存在缺陷的机器。错误的实现导致握手信息可以被重放。然而他们不能提供针对重放信息的攻击样例。我们猜想攻击者可以重放特定的信息欺骗站点重装一个 TLS 会话密钥，即密钥重载攻击存在可能性。我们认为能否实施实际的攻击在未来的工作里是很有趣的一点。

重用 IVs 在千疮百孔的 WEP 协议[17,18]里也存在问题。特别值得一提的是，Borisov 等人发现某个无线网卡每次初始化时都将 WEP IV 初始为零。因此，较小的 IVs 值产生的密钥流可能被重用[18]。然而，和密钥重载攻击不同的是，重置 IV 值不能被远程触发。

7.2 Wi-Fi 和网络协议安全

在对四次握手的一次正式分析中，何和 Mitchell 发现了一个拒绝服务漏洞[38,55]。这促使四次握手规范的改善[1]。2005 年，何等人提出了一个关于四次握手和组密钥握手的正确

性证明[39]。但是他们没有对密码选择和降级进行准确地建模。这使得 Vanhoef 和 Piessens 提出了针对四次握手的降级攻击[72]。在他们的攻击里，在传输 Message 3 时，AP 被欺骗使用 RC4 来加密组密钥。这种攻击只有在支持 WPA-TKIP 的网络里有效，因为 WPA-TKIP 显然是一个不安全的加密协议[66,69]。此外，在参考文献[39]中使用的模型没有定义何时安装协商的会话密钥或者传输的组密钥。 然而我们认为这一时机十分重要，有可能会导致密钥重载攻击。

FT 握手是基于四次握手的[5]，但是业界没有对它进行正式的安全性分析。现有的工作重点都是放在握手的性能上。例如参考文献[11,46]。

部分工作是研究协商主密钥（PMKs）的认证机制[19,21,59,75]。其中一些机制依赖于初始时创建一个安全的 TLS 会话[9]。因此，近期对 TLS 的攻击也会影响这些机制。例如[10,14,15,27,62]。本文中我们对协商主密钥进行研究，而把主要精力放在如何从协商主密钥或预共享主密钥中获取新的会话密钥。

Beck 和 Tews 在 WPA-TKIP 上实现了第一次关于数据保密协议的实际攻击[66]。他们展示了如何解密一个小 TKIP 数据包，恢复 MIC key，之后对数据包进行了伪造。他们的攻击进一步改善当前的一些工作[36,67,69,70]。研究人员还利用 RC4 的缺陷对由弱 key 构建的 TKIP 包进行攻击[6,57,71]。如今 TKIP 因为其安全问题已经被 Wi-Fi 协会所摒弃。

尽管 CCMP 收到一些批评[60]，但它已经被证明可以提供类似 OCB 模式的安全保障。在参考文献[31]中，Fouque 等人讨论了在 CCMP 中 nonces 被重用时消息伪造攻击的原理。

众所周知，GCM 密码在使用短验证标签和重用 nonces 时是脆弱的。Böck 等人实际研究了当 GCM 在 TLS 被使用时 nonce 重用问题，发现一些服务器确实存在 nonces 重用问题。我们的攻击在 802.11 上的 GCMP 和之前不同是因为当一个端点重用 nonce 时我们可以对其进行控制。并且 GCMP 在连接双方使用相同的认证 key。所以涉及到 GCM 的加密协议就会显得脆弱[35,56]。

最后，在 Wi-Fi 实现或者相关技术上的安全问题的一些工作值得被注意。例如在 Wi-Fi 保护设置（WPS）上发现了设计缺陷[73]，在驱动程序[13,20]发现了漏洞，路由器被发现可以使用预共享 key，等等。

8. 总结

尽管四次握手和组密钥交换握手都被证明为安全的，但我们展示出它们都存在被密钥重载攻击的可能。这些攻击方式和已经被证实的安全标准并不冲突，但是强调了它们所使用的模型的限制。特别的，模型没有指定何时应该装载一个用于数据保密协议的密钥。更多的，我们展示了 PeerKey 和 fast BSS Transition 握手同样会被密钥重载攻击所影响。

所有我们测试的 Wi-Fi 设备都会被针对组密钥握手的攻击所影响。这就允许攻击者重放广播和组播的数据帧。当四次握手或者 fast BSS Transition 握手被攻击的时候，这种影响范围取决于它们使用的哪一种数据保密协议。在所有情况下，都有可能解密数据帧，因此可以劫持 TCP 链接。这就允许攻击者向未加密的 HTTP 连接中注入数据。更多的，针对 Android 6.0，我们的攻击使其装载了一个全为零的密钥，这让所有的安全保障都失效了。

更令人担心的是，如果我们的握手信息由于背景噪音而丢失，密钥重载攻击可能会自动地进行。这意味着，在一定的条件下，即使没有攻击者，nonces 也可能被重用。

将来一个有趣的研究方向是其他协议的实现是否也会受到密钥重载攻击的影响。考虑到

消息可能丢失的协议可能会显得特别脆弱，毕竟这些协议用设计用来重传帧时，就有可能密钥重载。

致谢

这篇研究是由 KU Leuven 基金和 the imec High Impact Initiative Distributed Trust 赞助。

参考链接

https://papers.mathyvanhoef.com/ccs2017.pdf
https://www.krackattacks.com/