无线网络（WI-FI）保护协议标准 WPA2 漏洞

综合分析报告

安天安全研究与应急处理中心（Antiy CERT）

初稿完成时间：2017 年 10 月 17 日 22 时 08 分

首次发布时间：2017 年 10 月 23 日 12 时 00 分

本版更新时间：2017 年 10 月 23 日 12 时 00 分

扫二维码获取最新版报告

目 录

1 概述

欧洲鲁汶大学的博士后安全研究员 Mathy Vanhoef 在 10 月 15 日披露无线网络（Wi-Fi）保护协议标准 WPA2 的高危漏洞[1,2]。漏洞允许在 Wi-Fi 范围内的攻击者监听计算机和接入点之间的 Wi-Fi 流量。该漏洞影响协议本身，且对 WPA 和 WPA2 均有效，因此支持 WPA/WPA2 协议的软件或硬件均受到影响。

漏洞披露后，安天与炼石的工程师迅速响应，对该漏洞展开联合分析，形成此报告。

2 漏洞分析

WPA 全名为 Wi-Fi Protected Access，有 WPA 和 WPA2 两个标准，是一种保护无线网络（Wi-Fi）安全的协议[3]。WPA 实现了 IEEE 802.11i 标准的大部分，是在 802.11i 完备之前替代 WEP 的过渡方案，后被 WPA2 取代。由于 WPA 和 WPA2 均基于 802.11i，因此在技术层面上几乎是相同的，主要区别在于 WPA2 要求支持更安全的 CCMP。WPA 和 WPA2 均使用 802.11i 中定义的四次握手，客户端（Station, STA）和接入点（Access Point, AP）通过四次握手相互验证和协商名为成对临时密钥（Pairwise Transient Key, PTK）的会话密钥。PTK 通过成对主密钥（Pairwise Master Key, PMK）、AP 随机数 ANonce、STA 随机数 SNonce 和双方 MAC 地址等计算生成，其中 PMK 由登录密码等双方均已知的信息计算生成。而后续正常数据加密所使用的临时密钥（Temporal KEY, TK）即派生自 PTK。各密钥、参数的关系如下图所示。

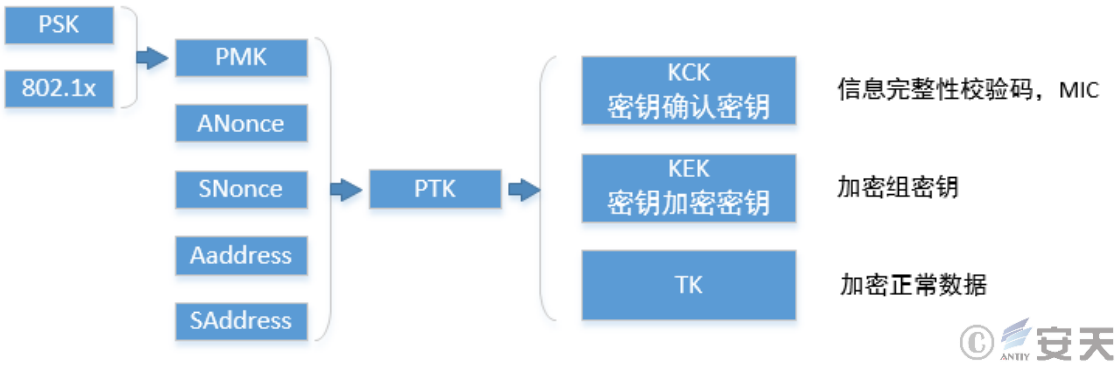

图 2-1 四次握手中各密钥、参数关系

四次握手的过程可概括如下：

1）AP 发送自己的随机数 ANonce 给 STA；

2）STA 生成随机数 SNonce，计算出 PTK，并将 SNonce 和信息完整性校验码 MIC 发送给 AP；

3）AP 收到 SNonce，计算出 PTK（此时双方都已有 PTK），将组密钥 GTK 加密后连同 MIC 发给 STA；

4）STA 收到 GTK，安装 PTK 和 GTK，发送 ACK 确认。AP 收到确认后安装 PTK。

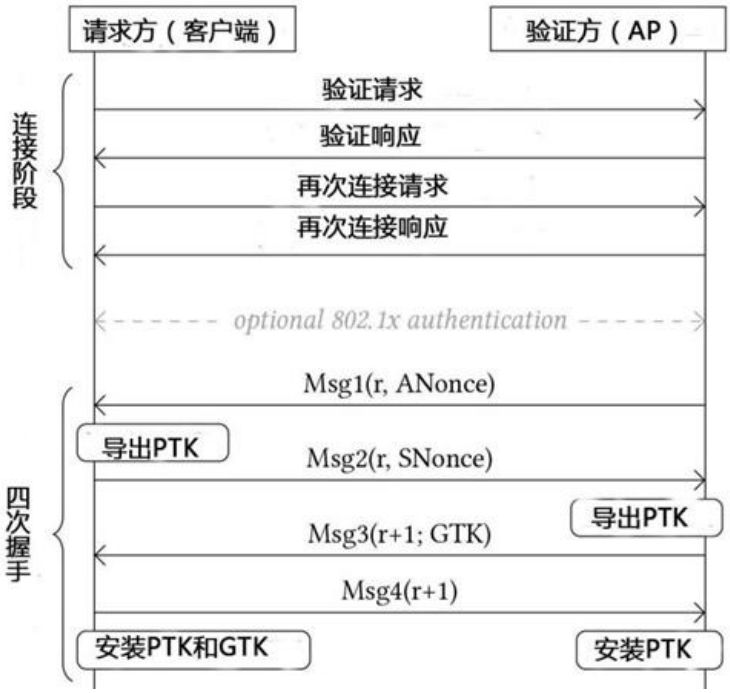

图 2-2 四次握手过程

Vanhoef 将披露的漏洞攻击命名为 KRACK（Key Reinstallation Attack），攻击者通过在 Wi-Fi 范围内利用中间人的手段对 WPA/WPA2 协议的四次握手交互验证的第三阶段进行攻击。此时受害者已经安装了密钥，通过中间人手段进行增量式重放攻击后，迫使受害者使用攻击之前的密钥流加密数据。鉴于 WPA/WPA2 为对称加密，因此可以通过简单的明文密文得到可重复利用的密钥流。攻击者一旦获得密钥流就可以对 Wi-Fi 流量数据解密。

2.1 利用原理

此漏洞利用的核心在于密钥重装，该模式基于 WPA/WPA2 协议中建立连接时的四次握手流程。在四次握手过程中，AP 和客户端将会协商一个用于加密接下来通信数据的加密密钥，客户端在收到 AP 发来的第三次握手的信息（消息 3）后会核实 MIC，若正确将会安装加密密钥 key，用于加密正常的数据帧，并向 AP 发送响应作为确认。根据协议规则，若 AP 无法正确接收到确认，将引发数据重传，重新发送消息 3。客户端每次接收到消息 3 时都会重装相同的会话密钥。攻击者可利用此次握手过程，暴力增量式发送消息 3，从而强制重置数据保密协议使用的增量传输数据包数（nonce）和接收重放计数器，导致密钥重用。攻击者可以通过这种方法重放、解密和/或伪造数据包。

KRACK 攻击可分为 4 种场景：

1）客户端（受害者）接受消息 3 的明文重传时的密钥重装攻击；

2）受害者仅接受消息 3 的加密重传时的密钥重装攻击；

3）对组密钥握手攻击即时密钥安装；

4）对组密钥握手攻击延迟密钥安装。

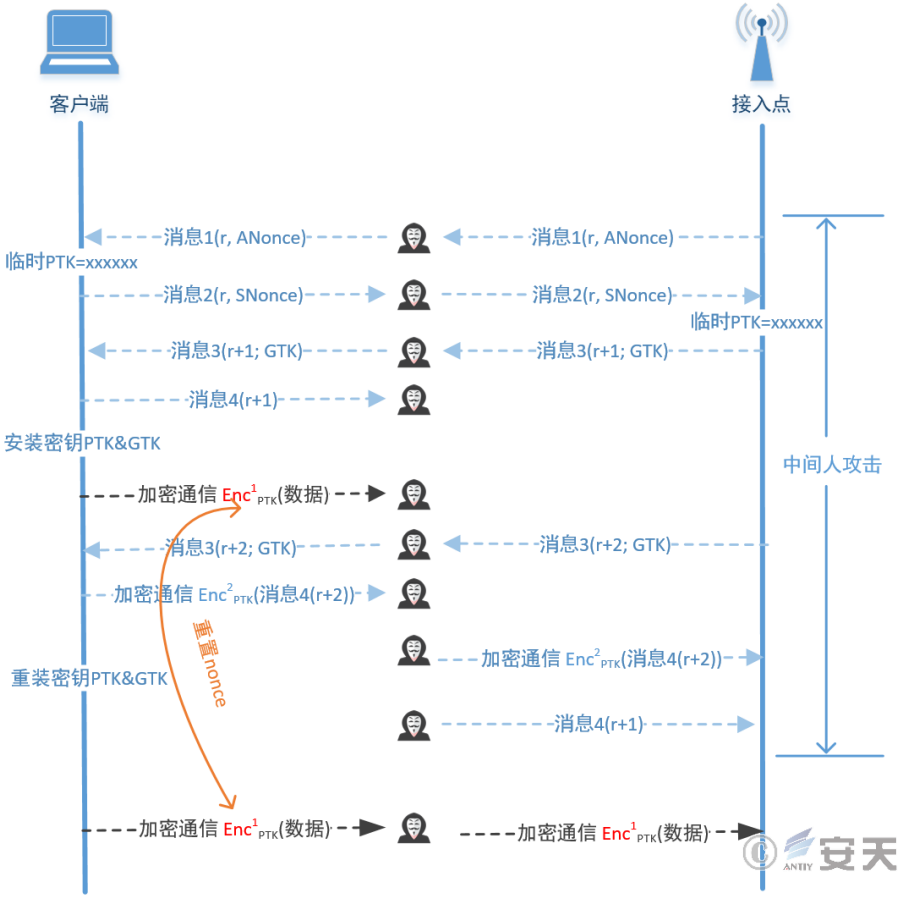

图 2-3 接受消息 3 明文重传时执行密钥重装攻击示意图

图 2-3 中展示了攻击者解密一个密文数据包的攻击流程。如果攻击者已知第一个发送的密文数据包的明文，则可以恢复出用于加密该明文数据的密钥流。由于四次握手协议的设计允许通过重传消息 3 来重装密钥 PTK 和 GTK，并重置待发送数据包的 nonce 值，导致客户端使用相同的密钥流加密下一个数据包，进而导致攻击者可以解密客户端发送的下一个密文数据包。

上述解密过程是基于攻击者知道第一个数据包明文这一前提，然而攻击者有时无法预知数据包中的全部字段值（比如可能存在的随机字段），所以攻击者可能需要多次重传消息 3 以收集更多数据来进行解密。图 2-3 中仅展示了恢复一个密文数据包的攻击过程，但是攻击者可以通过多次重传并合理选择重传时机（等待客户端在重传之前发送足够多的数据），甚至通过解除对客户端的认证来迫使客户端重新执行四次握手协议，实现解密多个数据包的目的。

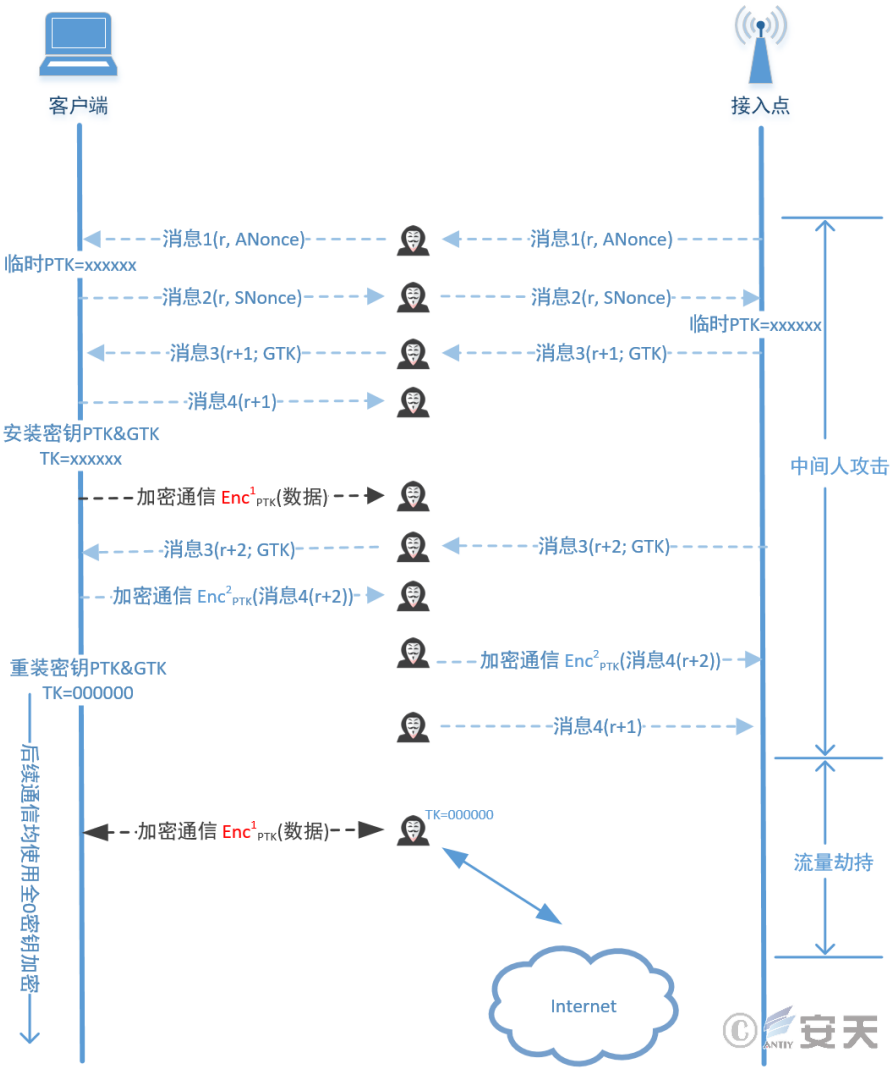

图 2-4 接受消息 3 明文重传时使用全 0 替换真实密钥攻击示意图

由于 2.4 和 2.5 版本的 wpa_supplicant（Android 6 及以上的版本）在收到重传的消息 3 后对密钥重装过程的错误实现，使得在客户端接收到攻击者重传的消息 3 后，数据加密密钥 TK 被置为全 0。这一错误实现极大的简化了密文恢复攻击，利用数据加密密钥为全 0 这个信息，攻击者不再需要已知明文条件就可以解密客户端发送的后续数据包。基于上述背景，攻击者通过中间人攻击迫使客户端改用全 0 的密钥之后，就可实现对客户端流量劫持，监控并篡改客户端发出的全部数据。

2.2 KRACK 攻击中 nonce 重用的密码分析

Nonce 重用引发的后果与所采用的数据保密协议密切相关。三种数据保密协议 TKIP、CCMP 和 GCMP 所采用的数据加密算法分别为流密码 RC4、认证加密算法 AES-CCM 和认证加密算法 AES-GCM，其中 AES-CCM 和 AES-GCM 的加密部分都是基于 CTR 模式的流式加密。即可认为 TKIP、CCMP 和 GCMP 三

种协议均采用流式加密，即明文数据与算法生成的密钥流按比特逐位异或后得到密文数据。流式加密的问题是在密钥固定的条件下重用 nonce 时总会生成相同的密钥流。这一特性可以被利用来解密数据包。

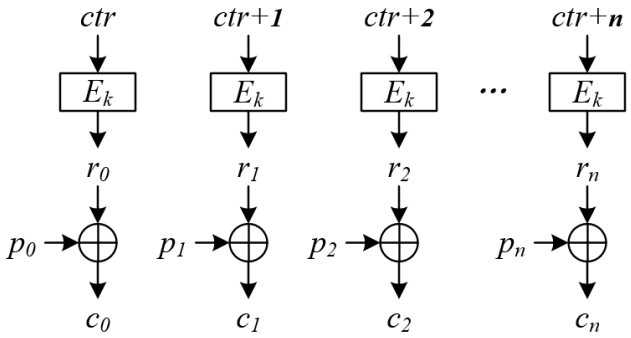

图 2-5 CTR 模式下加密过程示意图

上图展示了 CTR 模式下用密钥 k 和计数器 ctr 加密明文消息 $P = p_0 \| p_1 \| \cdots \| p_n$ 的过程，其中 $p_i, 0 \le i \le n$ 分别表示一个分组长度（采用 AES 算法时为 128 比特，明文消息长度并不一定是分组长度的整数倍）。注意图中没有展示 CCM 和 GCM 模式中应用 CTR 模式进行加密时所涉及到的一些细节。正确使用的情况下，计数器 ctr 的值不断累加（值不重复!!）并在算法和密钥的作用下生成具有强伪随机特性的密钥流 $r_0 \| r_1 \| \cdots \| r_n$，则对应的密文为

$$C = c_0 \| c_1 \| \cdots \| c_n = p_0 \oplus r_0 \| p_1 \oplus r_1 \| \cdots \| p_n \oplus r_n$$

TKIP 中基于 RC4 流密码的加密过程与此类似不再赘述。

在 KRACK 的攻击中，通过重放消息 3 能够达到迫使受害者重用 nonce，导致上述加密过程中相同计数器 ctr 值的重复出现。相同密钥相同计数器 ctr 条件下，生成的密钥流 $r_0 \| r_1 \| \cdots \| r_n$ 值相同，攻击者可据此解密数据包。用 $KeyStream$ 表示该密钥流，P_1 和 P_2 表示两组明文数据，假设 $KeyStream$，P_1，P_2 具有相同的比特长度，则两组明文对应的密文分别为：

$$C_1 = P_1 \wedge KeyStream$$
$$C_2 = P_2 \wedge KeyStream$$

其中 ^ 表示逐比特异或操作。攻击者可以通过网络收集到密文 C_1 和 C_2，如果攻击者知道密文 C_1 对应的明文 P_1，则可以据此恢复明文 P_2 的信息：

$$P_2 = C_2 \wedge keystream = C_2 \wedge (P_1 \wedge C_1)$$

在实际情况中，通常能找到已知内容的数据包，所以可以认为在密钥固定的条件下重用 nonce 时获得密文数据包可根据上述过程解密。即使已知内容的数据包确实无法获得，当对消息类型有足够的知识（比如消息为英文字符）的条件下，也可能解密还原出明文。**值得注意的是，虽然 nonce 重用会导致密文数据包被解密，但并不导致密钥 TK、PTK、PMK 以及 WiFi 登陆密码的泄露，因此 WPA2 的密码体系只是被绕过，并没有被攻破**。分组密码算法（AES）本身的安全性保证了即使在输入 $ctr \| ctr+1 \| \cdots \| ctr+n$ 和输出 $r_0 \| r_1 \| \cdots \| r_n$ 已知的条件下，加密密钥 k 也不会泄露。

TKIP、CCMP 和 GCMP 三种数据加密协议，在数据机密性之外，还提供数据完整性保护。而重用 nonce 在不同的数据加密条件下在数据完整性保护方面会带来不同等级的安全隐患。

使用 TKIP 协议时，当解密完整的 TKIP 数据包之后（包括 MIC 字段），攻击者可进一步攻击 Michael 算法获得相应的 MIC 密钥。这是由于 Michael 算法本身的脆弱性导致的，在给定明文数据以及 MIC 值的条件下，攻击者可恢复出 MIC 密钥。借助恢复出的 MIC 密钥，攻击者可伪造该数据传输方向上的数据帧（TKIP 在不同的数据传输方向上使用不同的 MIC 密钥）。

使用 CCMP 协议时，虽然有研究展示了在重用 nonce 条件下数据伪造攻击的可能性，但都是停留在理论层面的攻击，难以在实际中生成真正的伪造数据包，仅能执行重放攻击以及数据包解密。

使用 GCMP 协议时 nonce 重用导致的安全问题最为严重。nonce 重用使得攻击者能够恢复出 GCM 模式中的认证密钥（H），由于 GCMP 协议在数据传输的两个方向上使用相同的密钥进行数据保护，这赋予了攻击者在数据传输的两个方向上均可伪造数据包的能力。作为认证加密的一种工作模式，GCM 模式由 CTR 加密算法以及 GHASH 验证算法组合而成，其中 CTR 算法部分直接采用传递给 GCM 模式的密钥 k 进行加密，而 GHASH 运算所需的验证子密钥 H 是 AES 算法利用密钥 k 加密全 128 比特的全 0 明文得到的 128 比特密文值。法国密码学家 Joux 指出当 nonce 重用时，攻击者可恢复出验证子密钥 H 的值（注意从 H 的值无法推算出 GCM 的输入密钥，这是由 AES 算法本身的安全性保证的）。攻击者获得 H 的值之后，GCMP 所提供的数据完整性保护形同虚设，也因此攻击者可以伪造数据包。

总体来说，对于 TKIP 和 GCMP，KRACK 攻击影响极其严重，攻击者可以对数据包重放、解密和伪造。对于 CCMP，虽然攻击者不能伪造，但是基于 TCP/IP 协议的特点，只要攻击者能够获得序列号，攻击者就可以劫持 TCP 流并将恶意数据注入到其中，后果同样十分严重。

3 漏洞影响及响应情况

通过此漏洞可实现解密 Wi-Fi 流量数据、数据包重组、TCP 连接劫持、HTTP 内容注入等。KRACK 攻击是通用的，适用于连接或使用 WPA/WPA2 的 WiFi 网络的所有类型设备。针对个人和企业网络以及所使用的任何加密套件（WPA-TKIP，AES-CCMP 和 GCMP）都有效，包括 Android，iOS，Windows，Linux， MacOS，OpenBSD 以及嵌入式和物联网（IoT）设备。尤其是 Android 6 及以上的版本，研究人员表示可针对 Android 6 系统客户端 wpa_supplicant“完美攻击”，并且该攻击技术易于实现。根据安天移动安全统计数据，Android 设备中 Android 6 系统占比约为 41.9%。

由于此次漏洞影响较大，范围较广，供应商纷纷发起响应或发布补丁，部分供应商响应情况见下表：

表 3-1 部分供应商响应情况[4]

厂商名称	官方响应	注释	最近检查	最近更新
Apple	没有官方响应，通过 twitter 发布非官方响应。	苹果已确认 wpa2 漏洞，并在最新的 beta 版本 iOS、macOS、 tvOS 和 watchOS 中修复此漏洞。	2017-10-17	2017-10-17
Arch Linux	wpa_supplicant,hostapd	N/A	2017-10-16	2017-10-16
AVM	链接	正在调查此安全问题，并会在需要时发布更新。根据链接内容，EOM 和 EOS 产品也将更新	2017-10-17	2017-10-17
Cisco	链接	思科表示无线产品受到这些漏洞的影响	2017-10-16	2017-10-16
Debian	链接	添加修补程序来修复 WPA 协议漏洞（CVE-2017-13077，CVE-2017-13078，CVE-2017-13079，CVE-2017-13080，CVE-2017-13081，CVE-2017-13082，CVE-2017-13086，CVE-2017-13087，CVE-2017-13088）。	2017-10-16	2017-10-16
Fedora	链接	固定版本：可手动安装	2017-10-17	2017-10-17
FortiNet	链接	FortiAP 5.6.1 修补以下漏洞 CVE-2017-13077 CVE-2017-13078 CVE-2017-13079 CVE-2017-13080 CVE-2017-13081 CVE-2017-13082	2017-10-16	2017-10-16
FreeBSD Project	链接	基于 wpa_supplicant 2.5。 建议使用有线连接或安装 security / wpa_supplicant 端口或软件包，直到修补完成。	2017-10-16	2017-10-16
HPE Aruba	链接	N/A	2017-10-16	2017-10-16
Intel Corporation	链接	N/A	2017-10-16	2017-10-16
LineageOS	链接	N/A	2017-10-17	2017-10-17
Linux	补丁链接	wpa_supplicant 2.4 及以上版本受到影响。wpa_supplicant v2.6 版本受到影响。	2017-10-16	2017-10-16
Microsoft	补丁链接	单击链接时，请接受 EULA，然后再次单击该链接	2017-10-16	2017-10-16
Mikrotik	链接	上周发布了固定版本，所以如果经常升级设备，则不需要进一步的操作。	2017-10-16	2017-10-16
OpenBSD	链接	针对 OpenBSD 6.1 和 6.0，已经发布了无线堆栈的勘误补丁。状态转换错误可能导致重新安装旧的 WPA 密钥。通过 syspatch 程序可以获得 amd64 和 i386 平台的二进制更新。源代码补丁可以在相应的勘误页面找到。由于这会影响内核，因此在修补后将需要重启。	2017-10-16	2017-10-16
Red Hat, Inc.	此漏洞会影响 Red Hat Enterprise Linux 6 和 7	N/A	2017-10-16	2017-10-16

	LINK 提供的 wpa_supplicant 的版本，链接			
Sophos AP	链接	N/A	2017-10-17	2017-10-17
SUSE/openSUSE	链接	N/A	2017-10-16	2017-10-16
TP-Link	链接	N/A	2017-10-17	2017-10-17
Ubuntu	链接	在 Ubuntu 17.04，Ubuntu 16.04 LTS 和 Ubuntu 14.04 LTS 中，可以直接更新 wpasupplicant 和 hostapd。	2017-10-16	2017-10-16
WatchGuard	链接	2017 年 10 月 15 日 星期日：AP120,320,322,420：，版本 8.3.0-657，仅云模式。 2017 年 10 月 30 日，星期一：AP300：版本 2.0.0.9 ,AP100,102,200 ,版本 1.2.9.14，AP120,320,322,420：，版本 8.3.0-657，非云（GWC 模式）	2017-10-16	2017-10-16
WiFi Alliance	链接	用户应参考 Wi-Fi 设备供应商的网站或安全建议 ,以确定他们的设备是否已经受到影响，并且安装可用的更新。	2017-10-16	2017-10-16

注：

如表中链接无法打开，可以从参考资料 4 中获取。

我们对表中 Linux 对应的八个补丁进行了分析，下面展示了各个补丁的作用原理。

1. rebased-v2.6-0001-hostapd-Avoid-key-reinstallation-in-FT-handshake.patch

在握手阶段 Reassociation-Response 时避免将 TK 重新安装到驱动程序部分，防止密钥重装攻击。

```
+        * Skip this if the STA has already completed FT reassociation and the
+        * TK has been configured since the TX/RX PN must not be reset to 0 for
+        * the same key.
         */
-       if (!sta->added_unassoc)
+       if (!sta->added_unassoc &&
+           (!(sta->flags & WLAN_STA_AUTHORIZED) ||
+            !wpa_auth_sta_ft_tk_already_set(sta->wpa_sm))) {
                hostapd_drv_sta_remove(hapd, sta->addr);
+               wpa_auth_sm_event(sta->wpa_sm, WPA_DRV_STA_REMOVED);
+               set = 0;
+       }
```

另外，只有在 TK 确认已经被卸载时，才允许安装配置，且不允许重复安装(当且仅允许成功配置一次)

```
+       case WPA_DRV_STA_REMOVED:
+               sm->tk_already_set = FALSE;
+               return 0;
        }

 #ifdef CONFIG_IEEE80211R
@@ -3250,6 +3253,14 @@ int wpa_auth_sta_wpa_version(struct wpa_state_machine *sm)
 }


+int wpa_auth_sta_ft_tk_already_set(struct wpa_state_machine *sm)
+{
+       if (!sm || !wpa_key_mgmt_ft(sm->wpa_key_mgmt))
+               return 0;
+       return sm->tk_already_set;
+}
```

2. rebased-v2.6-0002-Prevent-reinstallation-of-an-already-in-use-group-ke.patch

跟踪当前正在使用的 GTK 和 IGTK，当接收到(可能重传)的第一次握手阶段的消息或 WNM-睡眠模式响应时，如果密钥已经在使用中，则不安装新的密钥。这阻止了攻击者欺骗客户端去重置或更改与组密钥关联的序列计数器的行为。

安装 igtk 时：

```
+#ifdef CONFIG_IEEE80211W
+static int wpa_supplicant_install_igtk(struct wpa_sm *sm,
+                                       const struct wpa_igtk_kde *igtk)
+{
+       size_t len = wpa_cipher_key_len(sm->mgmt_group_cipher);
+       u16 keyidx = WPA_GET_LE16(igtk->keyid);
+
+       /* Detect possible key reinstallation */
+       if (sm->igtk.igtk_len == len &&
+           os_memcmp(sm->igtk.igtk, igtk->igtk, sm->igtk.igtk_len) == 0) {
+               wpa_dbg(sm->ctx->msg_ctx, MSG_DEBUG,
+                       "WPA: Not reinstalling already in-use IGTK to the driver (keyidx=%d)",
+                       keyidx);
+               return 0;
+       }
+
```

WNM-睡眠模式响应时：

```
#ifdef CONFIG_IEEE80211W
        } else if (subelem_id == WNM_SLEEP_SUBELEM_IGTK) {
-               struct wpa_igtk_kde igd;
-               u16 keyidx;
-
-               os_memset(&igd, 0, sizeof(igd));
-               keylen = wpa_cipher_key_len(sm->mgmt_group_cipher);
-               os_memcpy(igd.keyid, buf + 2, 2);
-               os_memcpy(igd.pn, buf + 4, 6);
-
-               keyidx = WPA_GET_LE16(igd.keyid);
-               os_memcpy(igd.igtk, buf + 10, keylen);
-
-               wpa_hexdump_key(MSG_DEBUG, "Install IGTK (WNM SLEEP)",
-                               igd.igtk, keylen);
-               if (wpa_sm_set_key(sm, wpa_cipher_to_alg(sm->mgmt_group_cipher),
-                                  broadcast_ether_addr,
-                                  keyidx, 0, igd.pn, sizeof(igd.pn),
-                                  igd.igtk, keylen) < 0) {
-                       wpa_printf(MSG_DEBUG, "Failed to install the IGTK in "
-                                  "WNM mode");
-                       os_memset(&igd, 0, sizeof(igd));
+               const struct wpa_igtk_kde *igtk;
+
+               igtk = (const struct wpa_igtk_kde *) (buf + 2);
+               if (wpa_supplicant_install_igtk(sm, igtk) < 0)
                        return -1;
-               }
```

被删除

3. rebased-v2.6-0003-Extend-protection-of-GTK-IGTK-reinstallation-of-WNM-.patch

这个补丁追踪最后配置的 GTK / IGTK 值，分别与 EAPOL-Key 帧和 WNM-睡眠模式帧配合，因为当这两种不同的机制的 GTK / IGTK 发生变化时候，跟踪单个值不足以及时发现检测可能的密钥重新配置行为。

记录两种模式下的 GTK：

```
 static int wpa_supplicant_install_gtk(struct wpa_sm *sm,
                                       const struct wpa_gtk_data *gd,
-                                      const u8 *key_rsc)
+                                      const u8 *key_rsc, int wnm_sleep)
 {
        const u8 *_gtk = gd->gtk;
        u8 gtk_buf[32];

        /* Detect possible key reinstallation */
-       if (sm->gtk.gtk_len == (size_t) gd->gtk_len &&
-           os_memcmp(sm->gtk.gtk, gd->gtk, sm->gtk.gtk_len) == 0) {
+       if ((sm->gtk.gtk_len == (size_t) gd->gtk_len &&
+            os_memcmp(sm->gtk.gtk, gd->gtk, sm->gtk.gtk_len) == 0) ||
+           (sm->gtk_wnm_sleep.gtk_len == (size_t) gd->gtk_len &&
+            os_memcmp(sm->gtk_wnm_sleep.gtk, gd->gtk,
+                      sm->gtk_wnm_sleep.gtk_len) == 0)) {
                wpa_dbg(sm->ctx->msg_ctx, MSG_DEBUG,
                        "WPA: Not reinstalling already in-use GTK to the driver (keyidx=%d tx=%d len=%d)",
                        gd->keyidx, gd->tx, gd->gtk_len);
@@ -757,8 +760,14 @@ static int wpa_supplicant_install_gtk(struct wpa_sm *sm,
        }
        os_memset(gtk_buf, 0, sizeof(gtk_buf));

-       sm->gtk.gtk_len = gd->gtk_len;
-       os_memcpy(sm->gtk.gtk, gd->gtk, sm->gtk.gtk_len);
+       if (wnm_sleep) {
+               sm->gtk_wnm_sleep.gtk_len = gd->gtk_len;
+               os_memcpy(sm->gtk_wnm_sleep.gtk, gd->gtk,
+                         sm->gtk_wnm_sleep.gtk_len);
+       } else {
+               sm->gtk.gtk_len = gd->gtk_len;
+               os_memcpy(sm->gtk.gtk, gd->gtk, sm->gtk.gtk_len);
+       }

        return 0;
 }
```

记录两种模式下的 Igtk：

```
 static int wpa_supplicant_install_igtk(struct wpa_sm *sm,
-                                       const struct wpa_igtk_kde *igtk)
+                                       const struct wpa_igtk_kde *igtk,
+                                       int wnm_sleep)
 {
        size_t len = wpa_cipher_key_len(sm->mgmt_group_cipher);
        u16 keyidx = WPA_GET_LE16(igtk->keyid);

        /* Detect possible key reinstallation */
-       if (sm->igtk.igtk_len == len &&
-           os_memcmp(sm->igtk.igtk, igtk->igtk, sm->igtk.igtk_len) == 0) {
+       if ((sm->igtk.igtk_len == len &&
+            os_memcmp(sm->igtk.igtk, igtk->igtk, sm->igtk.igtk_len) == 0) ||
+           (sm->igtk_wnm_sleep.igtk_len == len &&
+            os_memcmp(sm->igtk_wnm_sleep.igtk, igtk->igtk,
+                      sm->igtk_wnm_sleep.igtk_len) == 0)) {
                wpa_dbg(sm->ctx->msg_ctx, MSG_DEBUG,
                        "WPA: Not reinstalling already in-use IGTK to the driver (keyidx=%d)",
                        keyidx);
@@ -900,8 +913,14 @@ static int wpa_supplicant_install_igtk(struct wpa_sm *sm,
                return -1;
        }

-       sm->igtk.igtk_len = len;
-       os_memcpy(sm->igtk.igtk, igtk->igtk, sm->igtk.igtk_len);
+       if (wnm_sleep) {
+               sm->igtk_wnm_sleep.igtk_len = len;
+               os_memcpy(sm->igtk_wnm_sleep.igtk, igtk->igtk,
+                         sm->igtk_wnm_sleep.igtk_len);
+       } else {
+               sm->igtk.igtk_len = len;
+               os_memcpy(sm->igtk.igtk, igtk->igtk, sm->igtk.igtk_len);
+       }

        return 0;
 }
```


4. rebased-v2.6-0004-Prevent-installation-of-an-all-zero-TK.patch

跟踪 PTK 是否已经安装到驱动程序，并且 TK 部分已经从内存中清除。 这样可以防止攻击者欺骗客户端来安装全零 TK。

```
@@ -615,7 +614,7 @@ static int wpa_supplicant_install_ptk(struct wpa_sm *sm,
        enum wpa_alg alg;
        const u8 *key_rsc;

-       if (!sm->tk_to_set) {
+       if (sm->ptk.installed) {
                wpa_dbg(sm->ctx->msg_ctx, MSG_DEBUG,
                        "WPA: Do not re-install same PTK to the driver");
                return 0;
@@ -659,7 +658,7 @@ static int wpa_supplicant_install_ptk(struct wpa_sm *sm,

        /* TK is not needed anymore in supplicant */
        os_memset(sm->ptk.tk, 0, WPA_TK_MAX_LEN);
-       sm->tk_to_set = 0;
+       sm->ptk.installed = 1;
```


5. rebased-v2.6-0005-Fix-PTK-rekeying-to-generate-a-new-ANonce.patch

用于 PTK rekeying 的授权状态机，会在随机数生成的时候绕过 authenication2 状态，而直接进入之 PKT-START 状态，因为此时无需再次确认 PMK，可能导致随机数不“随机”，或遭致其他问题。

针对此问题，当切换至 PTKSTART 状态时，便生成一个新的 ANonce。

```
+static int wpa_auth_sm_ptk_update(struct wpa_state_machine *sm)
+{
+       if (random_get_bytes(sm->ANonce, WPA_NONCE_LEN)) {
+               wpa_printf(MSG_ERROR,
+                          "WPA: Failed to get random data for ANonce");
+               sm->Disconnect = TRUE;
+               return -1;
+       }
+       wpa_hexdump(MSG_DEBUG, "WPA: Assign new ANonce", sm->ANonce,
+                   WPA_NONCE_LEN);
+       sm->TimeoutCtr = 0;
+       return 0;
+}
```


6. rebased-v2.6-0006-TDLS-Reject-TPK-TK-reconfiguration.patch

当成功配置 TPK-TK 后，禁止重新配置相同的参数至驱动。

```
+       if (peer->tk_set) {
+               /*
+                * This same TPK-TK has already been configured to the driver
+                * and this new configuration attempt (likely due to an
+                * unexpected retransmitted frame) would result in clearing
+                * the TX/RX sequence number which can break security, so must
+                * not allow that to happen.
+                */
+               wpa_printf(MSG_INFO, "TDLS: TPK-TK for the peer " MACSTR
+                          " has already been configured to the driver - do not reconfigure",
+                          MAC2STR(peer->addr));
+               return -1;
+       }
+
```



```
 static int wpa_tdls_process_tpk_m1(struct wpa_sm *sm, const u8 *src_addr,
                                    const u8 *buf, size_t len)
 {
@@ -2004,7 +2036,8 @@ skip_rsn:
        peer->rsnie_i_len = kde.rsn_ie_len;
        peer->cipher = cipher;

-       if (os_memcmp(peer->inonce, ftie->Snonce, WPA_NONCE_LEN) != 0) {
+       if (os_memcmp(peer->inonce, ftie->Snonce, WPA_NONCE_LEN) != 0 ||
+           !tdls_nonce_set(peer->inonce)) {
                /*
                 * There is no point in updating the RNonce for every obtained
                 * TPK M1 frame (e.g., retransmission due to timeout) with the
@@ -2020,6 +2053,7 @@ skip_rsn:
                                   "TDLS: Failed to get random data for responder nonce");
                goto error;
        }
+       peer->tk_set = 0; /* A new nonce results in a new TK */
        }
```


7. rebased-v2.6-0007-WNM-Ignore-WNM-Sleep-Mode-Response-without-pending-r.patch

如果 WNM-睡眠模式尚未使用，则忽略对应的 WNM-睡眠模式请求。这可以避免处理意外的重传数据帧。

```
@@ -260,7 +260,7 @@ static void ieee802_11_rx_wnmsleep_resp(struct wpa_supplicant *wpa_s,

        if (!wpa_s->wnmsleep_used) {
                wpa_printf(MSG_DEBUG,
-                          "WNM: Ignore WNM-Sleep Mode Response frame since WNM-Sleep Mode has not been used in this association");
+                          "WNM: Ignore WNM-Sleep Mode Response frame since WNM-Sleep Mode operation has not been requested");
                return;
        }

@@ -299,6 +299,8 @@ static void ieee802_11_rx_wnmsleep_resp(struct wpa_supplicant *wpa_s,
                return;
        }

+       wpa_s->wnmsleep_used = 0;
+
        if (wnmsleep_ie->status == WNM_STATUS_SLEEP_ACCEPT ||
            wnmsleep_ie->status == WNM_STATUS_SLEEP_EXIT_ACCEPT_GTK_UPDATE) {
                wpa_printf(MSG_DEBUG, "Successfully recv WNM-Sleep Response "
--
```


8. rebased-v2.6-0008-FT-Do-not-allow-multiple-Reassociation-Response-fram.patch

驱动部分除非在客户端明确请求一个新的连接时才会开启一个连接事件。不过，重新配置相同的成对密钥或组密钥会导致 nonce 被重用的问题，因此要进行额外的检查以避免恶意攻击的发生，包括因为某种环境因素导致意外收到重传数据包的情况。

```
@@ -153,6 +153,7 @@ static u8 * wpa_ft_gen_req_ies(struct wpa_sm *sm, size_t *len,
        u16 capab;

        sm->ft_completed = 0;
+       sm->ft_reassoc_completed = 0;

        buf_len = 2 + sizeof(struct rsn_mdie) + 2 + sizeof(struct rsn_ftie) +
                2 + sm->r0kh_id_len + ric_ies_len + 100;
@@ -681,6 +682,11 @@ int wpa_ft_validate_reassoc_resp(struct wpa_sm *sm, const u8 *ies,
                return -1;
        }

+       if (sm->ft_reassoc_completed) {
+               wpa_printf(MSG_DEBUG, "FT: Reassociation has already been completed for this FT protocol instance - ignore unexpected retransmission");
+               return 0;
+       }
+
        if (wpa_ft_parse_ies(ies, ies_len, &parse) < 0) {
                wpa_printf(MSG_DEBUG, "FT: Failed to parse IEs");
                return -1;
@@ -781,6 +787,8 @@ int wpa_ft_validate_reassoc_resp(struct wpa_sm *sm, const u8 *ies,
                return -1;
        }

+       sm->ft_reassoc_completed = 1;
+
        if (wpa_ft_process_gtk_subelem(sm, parse.gtk, parse.gtk_len) < 0)
                return -1;
```


4 结论

KRACK 漏洞利用主要针对 WPA/WPA2 的四次握手过程，没有利用 AP 接入点，而是针对客户端的攻击。因此，用户的路由器可能不需要更新。对于普通家庭用户，应多关注各终端设备厂商的安全公告，及时更新配置或打补丁，优先更新笔记本电脑和智能手机等客户端。

对该漏洞的利用并没有破坏密码体系本身，而是对实现过程进行了攻击，因此基本可以绕过所有的安全监控设备。利用该漏洞能够在一个良好实现的网络环境中，通过良好实现的 WiFi 打开攻击面，为后续攻击打开通路。

目前使用 WPA2 的大多数家庭和商业无线应用客户端升级相对容易，但对于数百万难以及时更新的 IoT 无线设备，可能造成巨大影响。请大家保持警惕，我们会持续关注相关事件并积极应对。

附录一：参考资料

[1] Mathy Vanhoef, Frank Piessens. Key Reinstallation Attacks.

https://www.krackattacks.com/

[2] Mathy Vanhoef, Frank Piessens. Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2.

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

[3] Wikipedia. WPA.

https://zh.wikipedia.org/wiki/WPA

[4] Kristate, Github. Vendor Response.

https://github.com/kristate/krackinfo#vendor-response-complete

附录二：关于安天

安天是专注于威胁检测防御技术的领导厂商。安天以提升用户应对网络空间威胁的核心能力、改善用户对威胁的认知为企业使命，依托自主先进的威胁检测引擎等系列核心技术和专家团队，为用户提供端点防护、流量监测、深度分析、威胁情报和态势感知等相关产品、解决方案与服务。

安天的监控预警能力覆盖全国、产品与服务辐射多个国家。安天将大数据分析、安全可视化等方面的技术与产品体系有效结合，以海量样本自动化分析平台延展分析师团队作业能力、缩短产品响应周期。安天结合多年积累的海量安全威胁知识库，综合应用大数据分析、安全可视化等方面经验，推出了可抵御各类已知和未知威胁的多样化解决方案。

全球超过一百家以上的著名安全厂商、IT 厂商选择安天作为检测能力合作伙伴，安天的威胁检测引擎目前已为全球近十万台网络设备和网络安全设备、超过八亿部移动终端设备提供安全防护，其中安天移动检测引擎是全球首个获得 AV-TEST 年度奖项的中国产品，并在国际权威认证机构 AV-C 的 2015 年度移动安全产品测评中，成为全球唯一两次检出率均为 100%的产品。

安天技术实力得到行业管理机构、客户和伙伴的认可，安天已连续五届蝉联国家级网络安全应急服务支撑单位资质，亦是中国国家信息安全漏洞库六家首批一级支撑单位之一。

安天是中国应急响应体系中重要的企业节点，在红色代码 II、口令蠕虫、震网、破壳、沙虫、方程式、白象、魔窟等重大安全事件中，安天提供了先发预警、深度分析或系统的解决方案。

安天实验室更多信息请访问： *http://www.antiy.com*（中文）

http://www.antiy.net（英文）

安天企业安全公司更多信息请访问： *http://www.antiy.cn*

安天移动安全公司（AVL TEAM）更多信息请访问： *http://www.avlsec.com*

附录三：关于炼石

炼石网络是一家专注于业务应用安全与数据安全领域的创新型服务商。公司首创基于委托式安全代理技术的 CASB 实现模式，自主研发出 CipherGateway 业务应用安全网关，能够在不改变应用系统的情况下，以适配的方式，将与业务紧密结合的安全机制集成到应用系统中，为应用提供近乎于内建的安全能力，让企业以前所未有的便捷方式丰富应用系统的安全功能，从源头应对业务应用安全与数据安全风险。

炼石网络更多信息请访问： *http://www.ciphergateway.com*

微信号：炼石网络 CipherGateway