Les risques liés à l'exploitation de failles de sécurité dans les applications mobiles

Yohan Billon, Grégory Clerc

Abstract: L'augmentation constante du nombre d'utilisateurs d'applications mobiles populaire et continue amène un risque d'infection accrue de celles-ci. En effet, plus une application est connue et utilisée, plus il y a un danger de probabilité que des personnes ayant des intentions malveillantes voulant accéder aux données des utilisateurs soient fortes. La sécurité et la maintenance des applications mobiles sont alors un enjeu de taille afin de garantir la durabilité et la prospérité de celles-ci. Mais rien ne dure indéfiniment, et des failles de sécurité sont trouvées de jour en jour. Dès lors que celles-ci sont inconnues de l'éditeur, elles peuvent, suivant leurs criticités, permettre une exploitation alarmante et néfaste des données des utilisateurs. Cet article définit l'importance d'une sécurité et maintenance solide de la part des éditeurs des applications mobiles à travers le projet Pegasus et notamment sur l'exploitation d'une faille de sécurité de l'application WhatsApp.

Mots-clés: NSO Group Technologies, Pegasus, malware, surveillance, vulnérabilités zero-day, collecte, zéros clique, iOS, WhatsApp.

Introduction

NSO Group Technologies est une entreprise technologique de sécurité informatique principalement connue pour son logiciel espion propriétaire Pegasus, logiciel capable de surveiller à distance les smartphones sans aucun clic.[1] (Timberg and Herbaut)
Citizen Lab et Lookout ont découvert que le lien télécharge un logiciel exploitant des vulnérabilités zero-day inconnues et non corrigées dans iOS (Bazaliy et al.). Il collecte ainsi toutes les communications et emplacements des iPhones ciblés. Des chercheurs de Google ont, sur des rapports de Lookout, découvert un malware Android nommé Chrysaor créé par NSO Group Technologies qui serait en lien avec Pegasus.
Le 13 mai 2019, Facebook annonce une vulnérabilité associée à tous ses produits WhatsApp. Cette vulnérabilité a été grandement exploitée (CVE-2019-3568) par le logiciel Pegasus sur iOS et Android. Elle provoque un débordement de tampon dans la pile VoIP de WhatsApp permettant ainsi l'exécution de code arbitraire à distance via une série spécialement conçue de paquets RTCP envoyés à un numéro de téléphone cible. (Facebook). L'importance d'une sécurité et rapidité de maintenance est alors capitale, car une vulnérabilité de cette ampleur atteint une criticité maximale et peut causer de lourd dégât, vu sur la CVE-2019-11931, où Jeff Bezos fut une victime et se fut volé un grand nombre de données privé, ayant pour objectif de lui faire pression (Kirchgaessner).
Cet article a pour objectif d'investiguer sur la faille de sécurité CVE-2019-3568, pour en comprendre sa structure, exploitation et résolution, ainsi que les dégâts qu'elle peut provoquer.

[1] La victime reçoit un lien piégé sur une application possédant une faille exploitable, lequel est exécuté sans aucune interaction, d'où l'appellation "zéros clique". (WhatsApp : CVE-2019-11931, permettant l'exécution de code à distance).

CVE-2019-3568

La faille de sécurité CVE-2019-3568 permet à un attaquant distant d'exécuter du code arbitraire sur le système cible. Elle existe en raison d'une erreur de limite dans la pile VoIP de WhatsApp lors du traitement des paquets RTCP. Un attaquant distant peut alors envoyer une série de paquets RTCP spécialement conçus à un numéro de téléphone cible, déclenchant un buffer overflow et permettant l'exécution du code arbitraire envoyé sur le périphérique cible. Entraînant une compromission complète du système.

Définitions

Afin de comprendre les vulnérabilités de cette faille de sécurité, il faut comprendre certaines définitions que nous allons détailler.

Buffer Overflow

Un buffer (ou en français : tampon) est une section séquentielle de mémoire allouée pour contenir n'importe quoi, d'un simple caractère à un tableau d'entiers. Un buffer overflow (ou en français : débordement de mémoire tampon), se produit lorsque plus de données sont placées dans la mémoire tampon (qui est de longueur fixe) que ce qu'elle peut gérer.
Les informations supplémentaires, doivent alors être stockées quelque part, et débordent alors dans l'espace mémoire adjacent, corrompant ou écrasant les données contenues dans cet espace. Ce débordement entraîne généralement un plantage du système, mais il crée également la possibilité pour un attaquant d'exécuter du code arbitraire ou de manipuler les erreurs de codage pour provoquer des actions malveillantes.

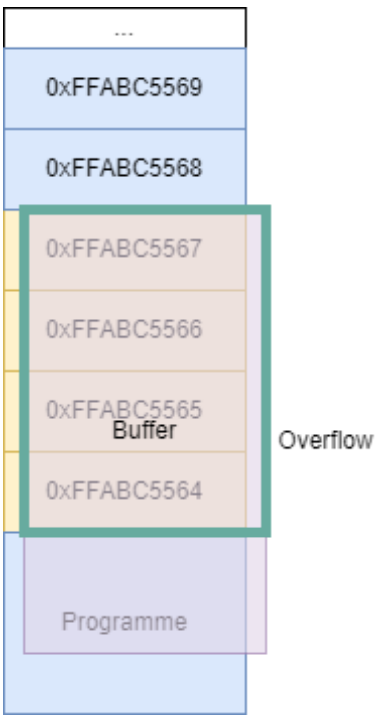


Schéma du buffer overflow

On peut voir sur ce schéma l'exemple du buffer overflow, qui modifie les données de la mémoire, écrasant ainsi des parties d'elle-même. Les données supplémentaires malveillantes peuvent contenir du code conçu pour déclencher des actions spécifiques, envoyant de nouvelles instructions à l'application attaquée qui pourraient entraîner un accès non autorisé au système.

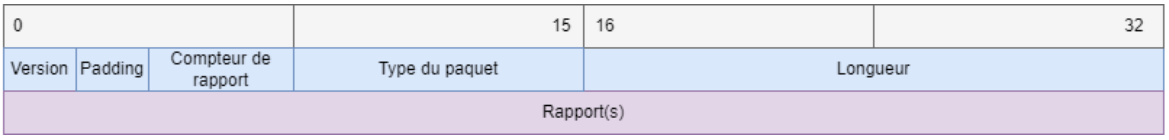
De nombreux langages de programmation sont sujets aux attaques par buffer overflow. Cependant, l'étendue de telles attaques varie en fonction du langage utilisé (Mayorov).

SRTCP

SRTCP est un profil de sécurité pour RTCP.

RTCP

RTCP (Real Time Control Protocol pour : Protocole de contrôle en temps réel), est un protocole de contrôle des flux RTP, permettant de véhiculer des informations basiques, comme sur les participants d'une session, et sur la qualité de service.



En-tête paquet RTCP (guill.net)

VoIP

Le VoIP (Voice over IP), est une technologie qui permet de transmettre la voix sur des réseaux compatibles IP (Internet, réseaux privés ou publics). Elle permet de gérer tous les flux multimédias (téléphonie, appels vidéo, messagerie instantanée et transferts de fichiers).

La voix et toutes les autres données voyagent par paquets sur des réseaux IP avec une capacité maximale fixe. Cette communication est moins fiable qu'un réseau téléphonique classique, car elle ne fournit pas de mécanisme basé sur le réseau pour garantir que les paquets de données ne sont pas perdus et livrés dans un ordre séquentiel, elle est donc plus sujet à la perte de données (en présence d'encombrement) que les circuits traditionnels à systèmes de commutation[2] (Cisco).

Sur des liaisons VoIP à trafic élevé, une latence peut être introduite dépassant les seuils autorisés. Cette forte latence provoque alors une charge excessive et provoque du bufferbloat (perte de paquets). Pour éviter les pertes indésirables de paquets qui arrivent dans le désordre le récepteur les re-séquence. Ce désordre, et ce, re-séquençage force le récepteur VoIP à stocker brièvement les paquets entrant dans une mémoire tampon de lecture (Cisco).

[2] Lors d'encombrement un système à commutation refusera toute nouvelles connexions tout en acheminant le reste sans dégradation. Sur un système VoIP la qualité des données en temps réel telles que les conversations téléphoniques se dégradent considérablement provoquant des latences et pertes.

Vulnérabilité

Vu précédemment, un certain nombre de langage sont sensible au buffer overflow, WhatsApp est écrit en Erlang (O'Connell), et dû à sa gestion de mémoire qui peut être gérée manuellement (*Erlang Run-Time System Application (ERTS)*), il en est sensible (Ce qui fut déjà le cas par le passé (ProHacktive)). De plus, Erlang permet l'utilisation de libraires provenant du C (notamment une librairie implémentant SRTCP pour les communications VoIP (Hancke)) qui en est lui-même aussi sensible.

Le protocole RTCP fournit des informations sur la façon dont le flux multimédia RTP se comporte, comme l'information de bufferbloat et en en-tête sa longueur (c.f: En-tête paquet RTCP).
L'information de bufferbloat peut être obtenue avec le paquet spécial SR (Sender Report: regroupe des statistiques concernant la transmission), notifiant alors le récepteur qu'il va y avoir des pertes (pourcentage de perte, nombre cumulé de paquets perdus, variation de délai, etc…) (guill.net). Ainsi, il est possible de simuler un délai extrêmement long forçant la mise en mémoire tampon de paquet ciblé.
Quant à la longueur, propriété de l'en-tête, servant notamment lors du dé-multiplexage des paquets composés par l'analyseur RTCP. Elle permet de valider le champ de longueur par rapport à la longueur du datagramme[3] et des octets restants dans le paquet. Dans le cas où cette vérification serait implémentée de manière incorrecte, il est alors possible d'envoyer et de faire valider un paquet plus gros.

Lorsqu'il y a une saturation sur les liaisons VoIP, la mémoire tampon accueille les paquets afin de libérer une partie de la bande passante. Ce trafic est transmis au-dessus de la couche UDP, ce qui implique qu'aucun mécanisme de contrôle de flux ou de retransmission des paquets perdus n'est offert par la couche transport. Cela implique que le récepteur VoIP doit calculer le taux de perte de paquets, et de réagir en conséquence au niveau de la couche applicative.

Pour effectuer l'attaque, l'attaquant crée une charge utile malveillante (série de paquets SRTCP) qui est ensuite encapsulée et mise sous forme de paquet d'initialisation d'appel VoIP. Ensuite, cette charge est envoyée via les serveurs de WhatsApp à l'appareil cible, qui remarque un appel WhatsApp normal. WhatsApp utilise E2EE, qui crypte l'appel pour tout le monde sauf les deux parties communicantes authentifiées. WhatsApp utilise E2EE uniquement pour la communication réelle des deux parties et non pour l'initialisation de la communication. C'est la première étape de cette défaillance, car cela permet aux attaquants de falsifier des messages qui, autrement, seraient supprimés au moment où ils ont été reçus, au lieu d'être ouverts.
Le deuxième point de défaillance, est le buffer overflow. Lors de la signalisation d'appel VoIP par WhatsApp, l'appareil du destinataire lance une procédure d'initialisation qui prépare le système à pouvoir gérer toutes les issues possibles de cet appel (prendre l'appel, refuser, etc.). Quelque part dans les paramètres d'appel envoyés à ce processus se trouve l'endroit où le débordement de la mémoire tampon est effectué. Téléchargeant et installant alors Pegasus même si le destinataire ne répond pas à l'appel (Filippidis).

[3] Paquet de données dans un réseau informatique utilisé par des protocoles orientés non connectés tel que : IPX ou UDP

Cas réel

Afin de voir comment ces vulnérabilités sont exploitées afin d'attaquer le système, nous utiliserons l'étude de Zimperium (Société privée de sécurité mobile).

Gestion des paquets SRTCP

Regardons la fonction de gestion des paquets SRTCP utilisée par WhatsApp :

```
714   __Asize = _Asize;
715   packetSize = brustStructure->totalPacketLenght;
716   if ( _Asize > (unsigned int)(0x7A120 - packetSize) )
717   {
718     if ( LoggingLevel >= 1 )
719       LogInfo("wa_transport.cc", " not enough space for buffer burst packet of length %d packets", _Asize);
720 LABEL_95:
721     result = 1LL;
722     goto LABEL_96;
723   }
724   if ( !(_options & 0x80000000) )
725   {
726     _dst = (void *)(brustStructure->dstBuffer + packetSize);
727     v29 = (__int64)&v12[32 * brustStructure->packet_count];
728     *(_QWORD *)v29 = _dst;
729     memcpy(_dst, _src, __Asize);
730     *(_QWORD *)(v29 + 8) = __Asize;
731     _dst2 = (void *)sub_100B6D9F4(brustStructure->qword38, _options);
732     *(_QWORD *)(v29 + 16) = _dst2;
733     memcpy(_dst2, _src2, _options);
734     *(_DWORD *)(v29 + 24) = _options;
735     v31 = brustStructure->totalPacketLenght + __Asize;
736     v32 = 4 - (v31 & 3);
737     if ( !(v31 & 3) )
738       v32 = 0;
739     brustStructure->totalPacketLenght = v32 + v31;
740     ++brustStructure->packet_count;
741     goto LABEL_95;
742   }
743   result = 0LL;
744 LABEL_96:
745   brustStructure->dword20B54 = 0;
746   return result;
```

Fonction en charge du traitement des paquets dans le cadre de la pile VoIP de WhatsApp

À la ligne 716, il y a un contrôle de taille avec les paramètres de l'appelant packetSize et _Asize. Si packetSize est plus grand que 0x7A120, cela provoque un dépassement d'entier et le bloc suivant est ignoré. Ce qui conduit au bloc où il y a deux appels à memcpy, pouvant alors provoquer par leur appelle, un buffer overflow par les rapports des paquets RTCP.

Gestion du trafic entrant

La deuxième fonction qui est utilisée pour effectuer une attaque est la fonction de gestion du trafic entrant de WhatsApp :

```
1  _QWORD *__fastcall Vuln_handle_incoming_traffic(_QWORD *inout_res, __int64 IN_arg2, signed int size, __int64 IN_arg4, int IN_arg5)
2  {
3    __int64 _IN_arg2; // x20
4    if ( !inout_res )
5      return inout_res;
6    _IN_arg2 = IN_arg2;
7    if ( !IN_arg2 )
8      return inout_res;
9    _IN_arg5 = IN_arg5;
10   _IN_arg4 = IN_arg4;
11   _size = size;
12   _inout_res = inout_res;
13   mutex_Foo((_QWORD *)inout_res[15439]);
14   mutex_Foo((_QWORD *)_inout_res[15436]);
15   mutex_Foo((_QWORD *)_inout_res[15438]);
16   inout_res = mutex_Foo((_QWORD *)_inout_res[15437]);
17   arg2Check = 0;
18   v17 = 0;
19   if ( (unsigned int)_size >= 8 )
20     arg2Check = (unsigned int)*(unsigned __int8 *)(_IN_arg2 + 1) - 0xC8 < 0xB;
21   v12 = _size;
22   if ( (_DWORD)_IN_arg4 && !*((_DWORD *)_inout_res + 0x8172) )
23   {...}
42 LABEL_17:
43   if ( arg2Check )
44   {
45     v13 = (void (__fastcall *)(_QWORD, __int64, __int64, _QWORD, __int64))_inout_res[2485];
46     if ( v13 )
47     {
48       v14 = (*(_DWORD *)_IN_arg2 >> 4) & 1;
49       LODWORD(v10) = bswap32(*(_DWORD *)(_IN_arg2 + 4));
50       v13(_inout_res[2483], _IN_arg2, v12, v14, v10);
51       sub_100BA5E3C(_IN_arg2, v12, &v17);
52       if ( v14 )
53         v15 = 12LL;
54       else
55         v15 = 5LL;
56       inout_res = (_QWORD *)sub_100C1D7A4(_inout_res, v15, &v17, 4LL);
57     }
58     else if ( _IN_arg5 && (*(_DWORD *)_IN_arg2 & 0xFE00) == 51200 )
59     {
60       inout_res = memcpy((char *)_inout_res + 0x209F4, (const void *)_IN_arg2, _size);
```

Fonction de gestion du trafic entrant

À la ligne 60, memcpy est appelé sur des arguments de l'appelant, la taille n'est pas vérifiée, ce qui pourrait conduire à un buffer overflow par l'injection des arguments données par l'appelant.

Fix

Afin de corriger ces erreurs, les équipe de WhatsApp, on sur la première fonctions tout simplement enlever le bloc pouvant injecter les rapport des paquets en mémoire:

```
794       }
795     }
796     if ( LoggingLevel >= 1 )
797     {
798       LODWORD(v29) = __rev16(v29);
799       LogInfo("wa_transport.cc", "Unknown STUN msg type: 0x%x", v29);
800     }
801     goto LABEL_69;
802   }
803 LABEL_76:
804   v6->dword20FFC = 0;
805   return 1LL;
806 }
```

Partie de la fonction en charge du traitement des paquets dans le cadre de la pile VoIP de WhatsApp avec le bloc d'injection mémoire supprimé

Et sur la deuxième fonction, ils ont rajouté une condition de longueur de mémoire afin d'éviter un débordement sur celle-ci.

```
else if ( (unsigned __int64)_size <= 0x5C8 && _IN_arg5 && (*(_DWORD *)_IN_arg2 & 0xFE00) == 0xC800 )
{
  inout_res = memcpy(&v9->gap209E0[0x14], (const void *)_IN_arg2, (size_t)_size);
  *(_DWORD *)&v9->gap209E0[0x5DC] = (_DWORD)_size;
}
```

Partie de la fonction de gestion du trafic entrant avec une condition supplémentaire pour éviter le débordement

Pegasus

À travers cet article, nous avons vu que WhatsApp a implémenté sa propre version du protocole SRTCP en code natif. Par des erreurs humaines, cette vulnérabilité fut exploitée, notamment par le projet Pegasus, utilisant cette vulnérabilité afin de pouvoir introduire le téléchargement de son application au sein du système en usant des droits données à WhatsApp.

Cette vulnérabilité se base sur des concepts connus et simples, mais qui nécessite des conditions et connaissances précises afin de pouvoir être mise en place, ce qui restreint grandement son apparition et exploitation. En effet, le projet Pegasus ne cible pas le grand public, mais des personnalités précises, afin de la garder secrète le plus longtemps possible. Mais en ciblant des personnalités qui ont une influence majeure, l'attaquant peut alors les faire chanter, prévoir leurs actions ou bien même connaître leur opinion politique (ce qui fut le cas par le passé en assassinant des personnalités contre les régimes en place).

Ce genre de vulnérabilité est alors extrêmement dangereux en termes de sécurité des données, l'enjeu est alors de pouvoir rapidement les localiser (avec par exemple des bug bounty) et les corriger dans les plus brefs délais, car une vulnérabilité bien exploitée peut provoquer des dégâts colossaux pouvant non seulement impacter la victime, mais aussi le grand public.

Les sociétés ont alors tout intérêt à mettre en place des systèmes permettant une grande sécurité pour éviter que ce genre de problème n'arrive, auquel cas elle pourrait perdre leur crédibilité et confiance des utilisateurs.

Travaux cités

Bazaliy, Max, et al. *Technical Analysis of Pegasus Spyware*. An Investigation Into Highly Sophisticated Espionage Software. 25 Août 2016. *Lookout pegasus technical analysis*, Lookout, https://info.lookout.com/rs/051-ESQ-475/images/lookout-pegasus-technical-analysis.pdf. Accessed 04 Janvier 2022.

Cisco. "Quality of Service for Voice over IP." *Cisco*, https://www.cisco.com/c/en/us/td/docs/ios/solutions_docs/qos_solutions/QoSVoIP/QoSVoIP.html. Accessed 5 January 2022.

Erlang Run-Time System Application (ERTS). Reference Manual. 12.2. *erts_alloc*, https://www.erlang.org/doc/man/erts_alloc.html. Accessed 5 Janvier 2022.

Facebook. *CVE-2019-3568*. 13 Mai 2019. *CVE-2019-3568*, https://www.facebook.com/security/advisories/cve-2019-3568.

Filippidis, Athanasios. "WhatsApp sues over a video-call vulnerability that targeted its users with spyware." *Medium*, 12 December 2019, https://medium.com/@thanasisflpd/whatsapp-sues-over-a-video-call-vulnerability-that-targeted-its-users-with-spyware-e6beb4c70462. Accessed 6 January 2022.

guill.net. "La voix sur IP : Les protocoles associés." *Voip : RTP et RTCP*, http://jmainy.free.fr/guill.web-/Voip5.html. Accessed 5 January 2022.

Hancke, Philipp. "webrtcH4cKS: ~ What's up with WhatsApp and WebRTC?" *webrtcHacks*, 22 April 2015, https://webrtchacks.com/whats-up-with-whatsapp-and-webrtc/. Accessed 5 January 2022.

Kirchgaessner, Stephanie. "Jeff Bezos hack: Amazon boss's phone 'hacked by Saudi crown prince.'" *The Guardian*, 21 January 2020, https://www.theguardian.com/technology/2020/jan/21/amazon-boss-jeff-bezoss-phone-hacked-by-saudi-crown-prince. Accessed 4 January 2022.

Mayorov, Alexandre. “Call Stack - buffer overflow vulnerability.” *ZeroBone*, 3 March 2021, https://zerobone.net/blog/cs/call-stack-buffer-overflow/. Accessed 5 January 2022.

O'Connell, Ainsley. “Inside Erlang, The Rare Programming Language Behind WhatsApp's Success.” *Fast Company*, 21 February 2014, https://www.fastcompany.com/3026758/inside-erlang-the-rare-programming-language-behind-whatsapps-success. Accessed 5 January 2022.

ProHacktive. “CVE-2016-10253.” *Prohacktive*, 18 Mars 2017, https://kb.prohacktive.io/index.php?action=detail&id=CVE-2016-10253&lang=en. Accessed 5 Janvier 2022.

Schulzrinne, Henning, et al. “rfc3550.” *IETF Tools*, Juillet 2003, https://datatracker.ietf.org/doc/html/rfc3550#section-6. Accessed 5 January 2022.

Timberg, Craig, and Guillaume Herbaut. “Apple iPhones were successfully hacked by NSO's Pegasus surveillance tool.” *The Washington Post*, 19 Juillet 2021, https://www.washingtonpost.com/technology/2021/07/19/apple-iphone-nso/. Accessed 4 Janvier 2022.