RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

300 SECONDES CHRONO : PRISE DE CONTRÔLE D'UN INFODIVERTISSEMENT AUTOMOBILE À DISTANCE

Philippe Trébuchet et Guillaume Bouffard
Agence nationale de la sécurité des systèmes d'information (ANSSI)
Laboratoire Architectures Matérielles et logicielles (**LAM**)

Les véhicules de plus en plus connectées

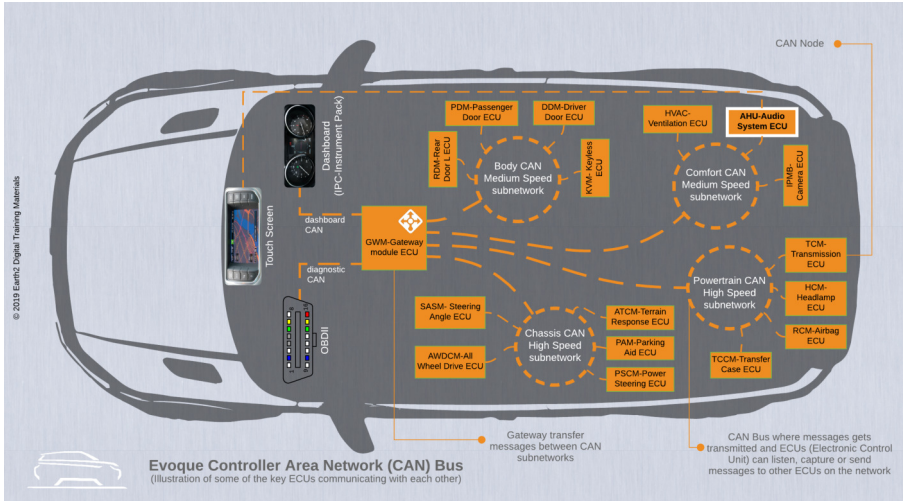
(image générée par ChatGPT-4o)

Les véhicules de plus en plus connectées

Une surface d'attaque de plus en plus grande !

(image générée par ChatGPT-4o)

Exemple d'architecture SI des véhicules connectés



(Source : `https://www.earth2.digital/blog/what-is-vehicle-can-bus-ecu-evoque-adam-ali.html`)

La sécurité, un risque pour la sureté des véhicules

- Plusieurs exemples d'exploitation à distance de véhicules connectés
  - Jeep Cherokee [BlackHat'15] (rappel 1,4M véhicules rappelés)
  - Tesla Model S [Tencent'16] (1ere étude)
  - BMW i3 [Tencent'18]...

[BlackHat'15] C. Miller and C. Valasek "*Remote Exploitation of an Unaltered Passenger Vehicle*" pdf
[Tencent'16] Car Hacking Research : Remote Attack Tesla Motors by Keen Security Lab. youtube
[Tencent'18] Experimental Security Assessment of BMW Cars : A Summary Report - Keen Security Lab, 2018 pdf

Modèle d'attaquant

Attaquant local :

- Présent à l'intérieur du véhicule
- Accès physique aux bus internes du véhicule

Modèle d'attaquant

Attaquant local :

- Présent à l'intérieur du véhicule
- Accès physique aux bus internes du véhicule

Attaquant distant :

- Possibilité d'interaction avec tous les capteurs externes
- Possibilité d'interaction avec tous les périphériques de communication sans fil.

Plateforme cible

- **Véhicule neuf de 2020**
  - **Caractéristiques techniques** :
    - services connectés
    - équipements principaux retenus dans le véhicule acheté : Aide au stationnement (capteurs de proximité avant et arrière), capteur de pression des pneus, **Bluetooth**, **Wi-Fi**, capteurs de luminosité, **aide au freinage d'urgence**
  - Calculateurs embarqués (ECU) associés d'intérêt pour l'étude :
    - calculateur airbag
    - calculateur télématique
    - calculateur ABS
    - calculateur Frein de Parking
    - calculateur climatisation
    - calculateur Colonne de direction
    - calculateur Unité CTL Habitacle
    - calculateur Injection
    - clé/Carte de démarrage

Architecture logicielle de l'infodivertissement

Interfaces

- Wi-Fi
- GSM
- Bluetooth

Architecture logicielle de l'infodivertissement

Interfaces

- Wi-Fi
- GSM
- Bluetooth

Fonctions de sécurité

- **ASLR (*Address Space Layout Randomization*)**
  - randomise les emplacements mémoire (code, pile, bibliothèques).
  - rend plus difficile les attaques ROP/JOP.
  - dépend fortement de l'entropie disponible.
- **W⊕X (*Data Execution Prevention* – DEP)**
  - interdit l'exécution de données en mémoire (ex. pile, tas).
  - empêche l'exécution de shellcodes injectés.

Architecture logicielle de l'infodivertissement

Interfaces

- Wi-Fi : desactivé par défaut.
- GSM : traffic non maîtrisé.
- Bluetooth

Fonctions de sécurité

- **ASLR (*Address Space Layout Randomization*)**
  - randomise les emplacements mémoire (code, pile, bibliothèques).
  - rend plus difficile les attaques ROP/JOP.
  - dépend fortement de l'entropie disponible.
- **W⊕X (*Data Execution Prevention* – DEP)**
  - interdit l'exécution de données en mémoire (ex. pile, tas).
  - empêche l'exécution de shellcodes injectés.

Architecture logicielle de l'infodivertissement

Interfaces

- Wi-Fi : desactivé par défaut.
- GSM : traffic non maîtrisé.
- **Bluetooth**

Fonctions de sécurité

- **ASLR (*Address Space Layout Randomization*)**
  - randomise les emplacements mémoire (code, pile, bibliothèques).
  - rend plus difficile les attaques ROP/JOP.
  - dépend fortement de l'entropie disponible.
- **W⊕X (*Data Execution Prevention* – DEP)**
  - interdit l'exécution de données en mémoire (ex. pile, tas).
  - empêche l'exécution de shellcodes injectés.

Architecture logicielle de l'infodivertissement

Interfaces

- Wi-Fi : desactivé par défaut.
- GSM : traffic non maîtrisé.
- **Bluetooth** déporté en dehors du noyau : `BluegoService` (service Java)
  - JNI et `libbluego.so`.
  - une seule CVE publique sur cette bibliothèque (`CVE-2018-20378`).

Fonctions de sécurité

- **ASLR (*Address Space Layout Randomization*)**
  - randomise les emplacements mémoire (code, pile, bibliothèques).
  - rend plus difficile les attaques ROP/JOP.
  - dépend fortement de l'entropie disponible.
- **W⊕X (*Data Execution Prevention* – DEP)**
  - interdit l'exécution de données en mémoire (ex. pile, tas).
  - empêche l'exécution de shellcodes injectés.

Libbluego : CVE-2018-20378

`libbluego` présente dans sa version 2.X dans l'ECU étudié.

- CVE publiée le 29/03/2019 sous la référence `CVE-2018-20378`
- MTU invalide dans la couche L2CAP.
- Cible : OpenSynergy Blue SDK[a] version **3.2** à 6.0.
  - … mais fonctionnel sur la version 2.X (`2.6.0-rc1`) présente dans l'ECU étudié
- Bibliothèque à source fermée (d'où le score CVSS relativement faible)

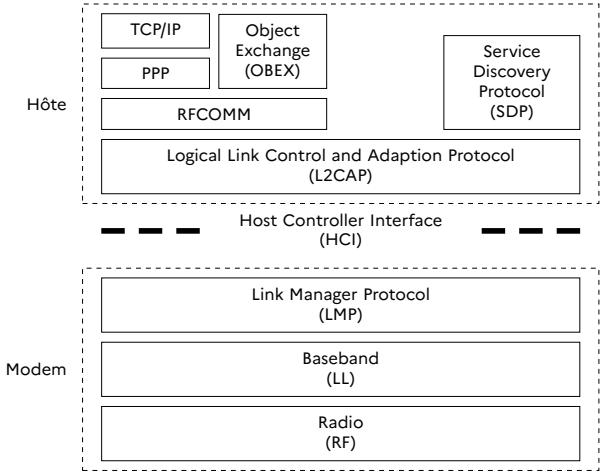
a. **Voir : `https://www.opensynergy.com/blue-sdk/`.**

"The L2CAP signaling channel implementation and SDP server implementation in OpenSynergy Blue SDK 3.2 through 6.0 allow remote, unauthenticated attackers to execute arbitrary code or cause a denial of service via malicious L2CAP configuration requests, in conjunction with crafted SDP communication over maliciously configured L2CAP channels. The attacker must have connectivity over the Bluetooth physical layer, and must be able to send raw L2CAP frames."

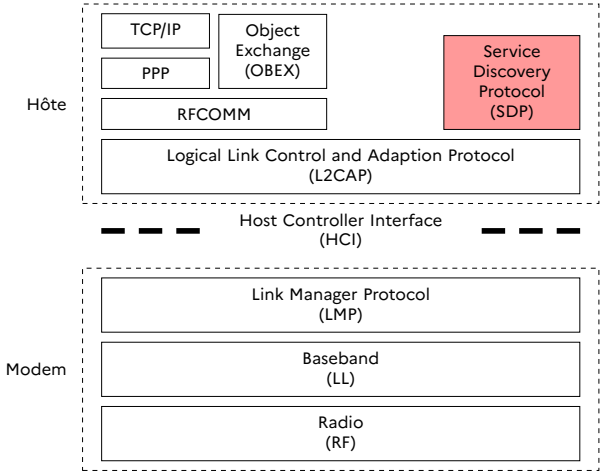
CVSS scores for CVE-2018-20378

Base Score	Base Severity	CVSS Vector	Exploitability Score	Impact Score	Source
5.4	MEDIUM	AV:A/AC:M/Au:N/C:P/I:P/A:P	5.5	6.4	nvd@nist.gov
7.5	HIGH	CVSS:3.0/AV:A/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H	1.6	5.9	nvd@nist.gov

[une partie de] La pile Bluetooth

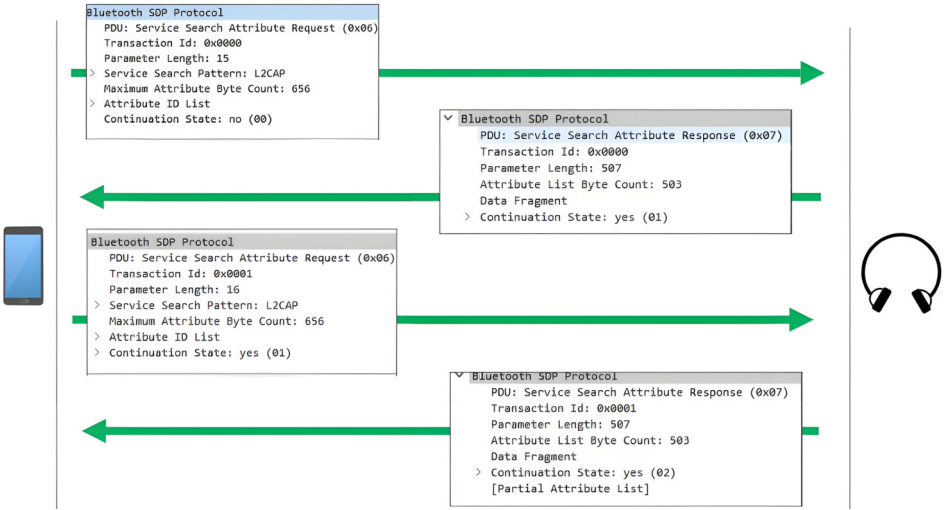


[une partie de] La pile Bluetooth



Protocole utilisable avant l'appairage

Service Discovery Protocol (SDP)



Libbluego : Récapitulatif CVE-2018-20378

Analyse de la CVE-2018-20378

- Les auteurs *prétendent* avoir réalisé une preuve de concept en faisant du JOP *(Jump Oriented Programming)*.
- Aucun détails de l'exploitation n'est public, même dans les publications afférentes !

Libbluego : Récapitulatif CVE-2018-20378

Analyse de la CVE-2018-20378

- Les auteurs *prétendent* avoir réalisé une preuve de concept en faisant du JOP *(Jump Oriented Programming)*.
- Aucun détails de l'exploitation n'est public, même dans les publications afférentes !

- **Cette CVE peut-elle être transposée sur le véhicule de cible ?**

Libbluego : Récapitulatif CVE-2018-20378

Analyse de la CVE-2018-20378

- Les auteurs *prétendent* avoir réalisé une preuve de concept en faisant du JOP *(Jump Oriented Programming)*.
- Aucun détails de l'exploitation n'est public, même dans les publications afférentes !

- **Cette CVE peut-elle être transposée sur le véhicule de cible ? Oui !**
  - … mais c'est loin d'être immédiat
  - … **les détails de la CVE ne s'appliquent pas à la cible !**
    - (car la cible embarque une version antérieur à la plus vieille version vulnérable)

Libbluego : Récapitulatif CVE-2018-20378

Analyse de la CVE-2018-20378

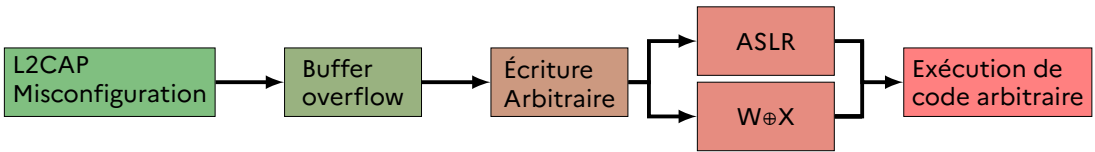
- Les auteurs *prétendent* avoir réalisé une preuve de concept en faisant du JOP *(Jump Oriented Programming)*.
- Aucun détails de l'exploitation n'est public, même dans les publications afférentes !

- **Cette CVE peut-elle être transposée sur le véhicule de cible ? Oui !**
  - … mais c'est loin d'être immédiat
  - … **les détails de la CVE ne s'appliquent pas à la cible !**
    - (car la cible embarque une version antérieur à la plus vieille version vulnérable)
  - **l'exploitation réalisée s'avère complètement différente** de celle **suggérée** par les auteurs

Vue générale du scénario d'attaque retenu

L2CAP Misconfiguration ⇢ Exécution de code arbitraire

Vue générale du scénario d'attaque retenu

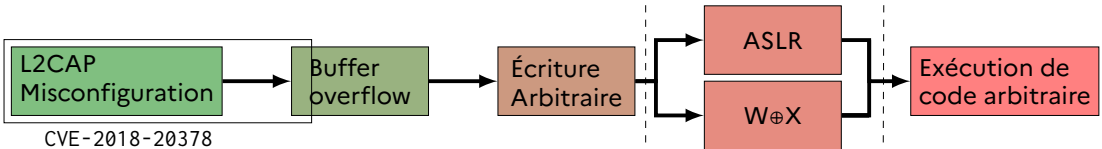


Vue générale du scénario d'attaque retenu

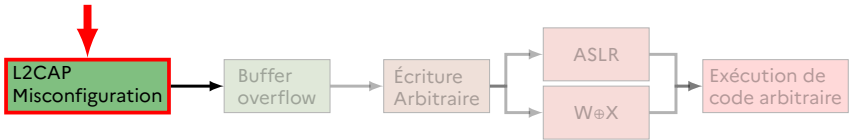
Étape 1
Exploitation du buffer overflow

Étape 2
Contournement des mécanismes de sécurité

Étape 3
Exécution de code arbitraire



Étape 1 / Confusion dans la configuration L2CAP

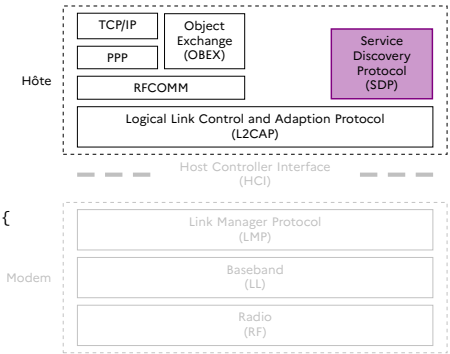


Objectif :

Transposer l'overflow présenté dans la `CVE-2018-20378`.

Étape 1 / CVE-2018-20378 : Bug dans la machine à état L2CAP

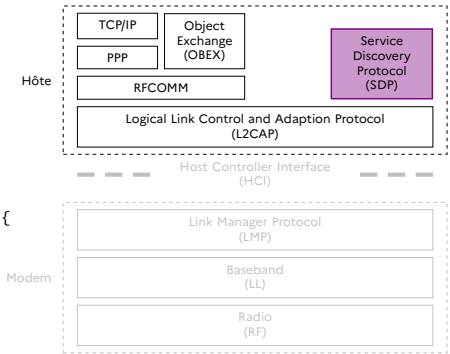
```
1  // Dans le fichier core/stack/l2cap/l2cap_sm.c
2  void L2Cap_HandleConfigReq(/* ... */) {
3   if (config_option[1] == configOption_MTU) {
4    in_mtu = LEtoHost16(data_iter + 2);
5    channel->MTU = in_mtu;
6    if (channel->ptProtocol->minimum_protocolMTU /* = 48 */ > in_mtu) {
7     l2cap_set_channel_as_invalid(channel);
8  }}}
```



- Pour SDP, la valeur minimale de la MTU est de 48 octets.

Étape 1 / CVE-2018-20378 : Bug dans la machine à état L2CAP

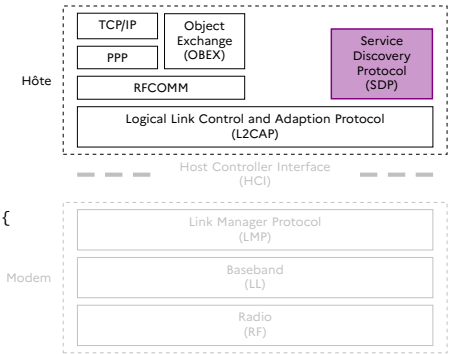
```
1  // Dans le fichier core/stack/l2cap/l2cap_sm.c
2  void L2Cap_HandleConfigReq(/* ... */) {
3   if (config_option[1] == configOption_MTU) {
4    in_mtu = LEtoHost16(data_iter + 2);
5    channel->MTU = in_mtu;
6    if (channel->ptProtocol->minimum_protocolMTU /* = 48 */ > in_mtu) {
7     l2cap_set_channel_as_invalid(channel);
8  }}}
```



- Pour SDP, la valeur minimale de la MTU est de 48 octets.
- Affectation avant vérification

Étape 1 / CVE-2018-20378 : Bug dans la machine à état L2CAP

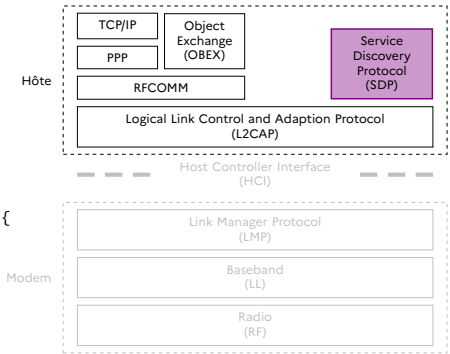
```
1  // Dans le fichier core/stack/l2cap/l2cap_sm.c
2  void L2Cap_HandleConfigReq(/* ... */) {
3   if (config_option[1] == configOption_MTU) {
4    in_mtu = LEtoHost16(data_iter + 2);
5    channel->MTU = in_mtu;
6    if (channel->ptProtocol->minimum_protocolMTU /* = 48 */ > in_mtu) {
7     l2cap_set_channel_as_invalid(channel);
8  }}}
```



- Pour SDP, la valeur minimale de la MTU est de 48 octets.
- Affectation avant vérification

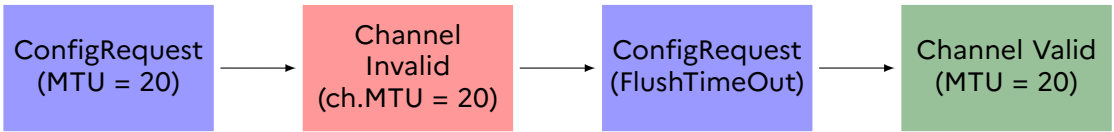
Étape 1 / CVE-2018-20378 : Bug dans la machine à état L2CAP

```
1  // Dans le fichier core/stack/l2cap/l2cap_sm.c
2  void L2Cap_HandleConfigReq(/* ... */) {
3   if (config_option[1] == configOption_MTU) {
4    in_mtu = LEtoHost16(data_iter + 2);
5    channel->MTU = in_mtu;
6    if (channel->ptProtocol->minimum_protocolMTU /* = 48 */ > in_mtu) {
7     l2cap_set_channel_as_invalid(channel);
8  }}}
```

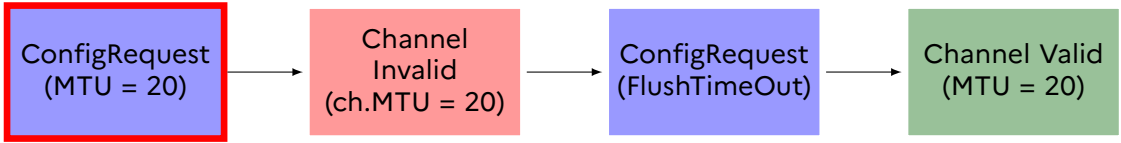


- Pour SDP, la valeur minimale de la MTU est de 48 octets.
- Affectation avant vérification
- Le **canal n'est pas fermé** quand il devient invalide !

Étape 1 / Confusion dans la configuration L2CAP

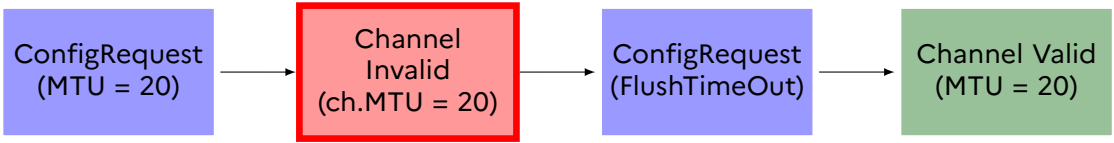


Étape 1 / Confusion dans la configuration L2CAP



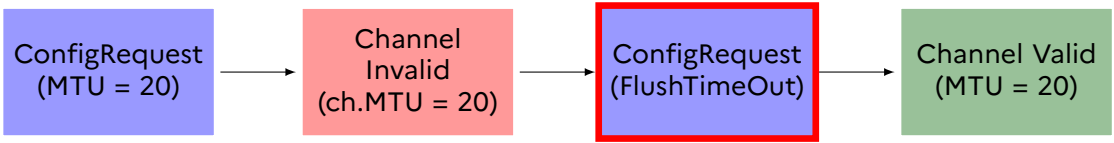

```
Frame 29: 21 bytes on wire (168 bits), 21 bytes captured (168 bits) on interface bluetooth0, id 0
// ...
Bluetooth L2CAP Protocol
    Length: 12
    CID: L2CAP Signaling Channel (0x0001)
    Command: Configure Request
        Command Code: Configure Request (0x04)
        Command Identifier: 0x02
        Command Length: 8
        Destination CID: Dynamically Allocated Channel (0x0040)
        0000 0000 0000 000. = Reserved: 0x0000
        .... .... .... ...0 = Continuation Flag: False
        Option: MTU
            Type: Maximum Transmission Unit (0x01)
            Length: 2
            MTU: 20
```

Étape 1 / Confusion dans la configuration L2CAP



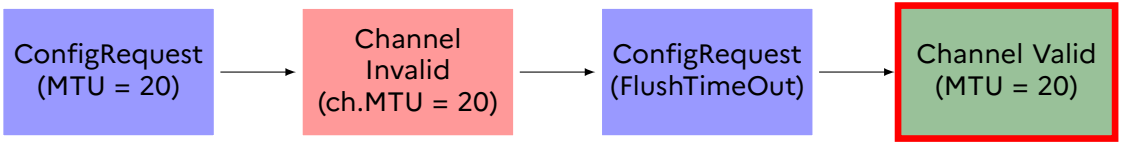

```
Frame 31: 23 bytes on wire (184 bits), 23 bytes captured (184 bits) on interface bluetooth0, id 0
// ...
Bluetooth L2CAP Protocol
    Length: 14
    CID: L2CAP Signaling Channel (0x0001)
    Command: Configure Response
        Command Code: Configure Response (0x05)
        Command Identifier: 0x02
        Command Length: 10
        Source CID: Dynamically Allocated Channel (0x0080)
        0000 0000 0000 000. = Reserved: 0x0000
        .... .... .... ...0 = Continuation Flag: False
        Result: Failure - unacceptable parameters (0x0001)
        Option: MTU
            Type: Maximum Transmission Unit (0x01)
            Length: 2
            MTU: 48
```

Étape 1 / Confusion dans la configuration L2CAP




```
Frame 34: 21 bytes on wire (168 bits), 21 bytes captured (168 bits) on interface bluetooth0, id 0
// ...
Bluetooth L2CAP Protocol
    Length: 12
    CID: L2CAP Signaling Channel (0x0001)
    Command: Configure Request
        Command Code: Configure Request (0x04)
        Command Identifier: 0x03
        Command Length: 8
        Destination CID: Dynamically Allocated Channel (0x0040)
        0000 0000 0000 000. = Reserved: 0x0000
        .... .... .... ...0 = Continuation Flag: False
        Option: Flush Timeout
            Type: Flush Timeout (0x02)
            Length: 2
            Flush Timeout (ms): 0x1337
```

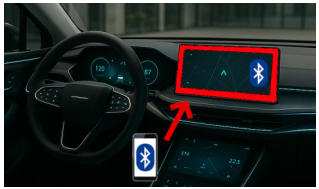
Étape 1 / Confusion dans la configuration L2CAP

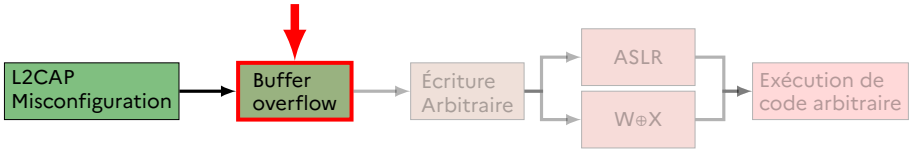



```
Frame 36: 27 bytes on wire (216 bits), 27 bytes captured (216 bits) on interface bluetooth0, id 0
// ...
Bluetooth L2CAP Protocol
    Length: 18
    CID: L2CAP Signaling Channel (0x0001)
    Command: Configure Response
        Command Code: Configure Response (0x05)
        Command Identifier: 0x03
        Command Length: 14
        Source CID: Dynamically Allocated Channel (0x0080)
        0000 0000 0000 000. = Reserved: 0x0000
        .... .... .... ...0 = Continuation Flag: False
        Option: MTU
            Type: Maximum Transmission Unit (0x01)
            Length: 2
            MTU: 20
```

```
Option: Flush Timeout
    Type: Flush Timeout (0x02)
    Length: 2
    Flush Timeout (ms): 0x1337
```

Étape 1 / Étude de l'overflow

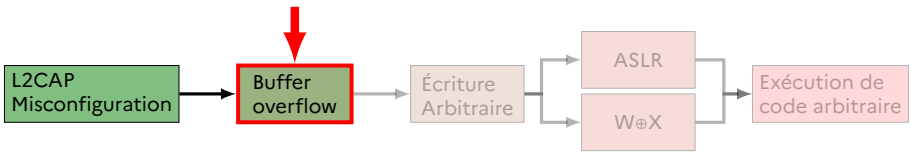




Objectif :

Exploiter le buffer overflow.

Étape 1 / Étude de l'overflow



Reverse :

(La valeur de la `mtu` correspond à la taille max d'un packet SDP.)

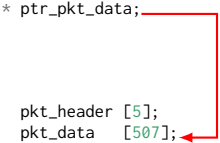
```
1  void SdpServHandleServiceSearchAttribReq ( /* ... */ ) {
2      // ...
3      mtu = MIN ( L2CAP_GetTxMtu(remdev->sdpServInfo->channelID), 0x200 );
4      size_resp = (mtu - 9) & 0xFFFF;
5      size_resp = MIN ( rx_pkt->MaximunAttributeByteCount, size_resp )
6      // ...
7  }
```

Étape 1 / Un overflow vers où ?

```
struct sdpServInfo {
  /* 0x0000 */  void    * next;
  /* 0x0004 */  void    * prev;
  /* 0x0008 */  uint8_t * ptr_pkt_data;
      // ...
      // ...
      // ...
      // ...
  /* 0x0088 */  uint8_t   pkt_header [5];
  /* 0x008D */  uint8_t   pkt_data   [507];
}; // taille = 648 (0x288) octets
```

Étape 1 / Un overflow vers où ?

```
struct sdpServInfo {
  /* 0x0000 */  void    * next;
  /* 0x0004 */  void    * prev;
  /* 0x0008 */  uint8_t * ptr_pkt_data;
      // ...
      // ...
      // ...
      // ...
  /* 0x0088 */  uint8_t   pkt_header [5];
  /* 0x008D */  uint8_t   pkt_data   [507];
}; // taille = 648 (0x288) octets
```



Étape 1 / Un overflow vers où ?

```
struct sdpServInfo {
  /* 0x0000 */  void    * next;
  /* 0x0004 */  void    * prev;
  /* 0x0008 */  uint8_t * ptr_pkt_data;
      // ...
      // ...
      // ...
      // ...
  /* 0x0088 */  uint8_t   pkt_header [5];
  /* 0x008D */  uint8_t   pkt_data   [507];
}; // taille = 648 (0x288) octets
```

- Un tableau statique de 7 `struct sdpServInfo` dans le segment bss.

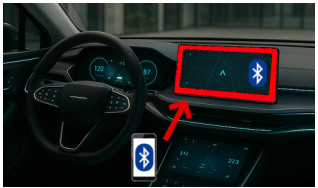
Étape 1 / Un overflow vers où ?

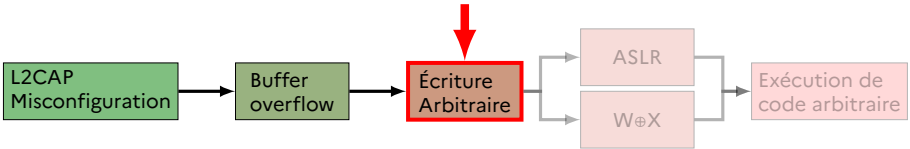
```
// bt.sdpServ {{{
struct sdpServInfo[0] { /* ... */ };
// ...
struct sdpServInfo[6] {
  // ...
  /* 0x0FB8 */  uint8_t   pkt_header [5];
  /* 0x0FBD */  uint8_t   pkt_data   [507];
};
  /* 0x11D6 */  uint32_t  SDPServ_Semaphore;
// }}}

// bt.sdpClient {{{
struct Protocol {
  /* 0x11DA */  protocol_callback * callback; // => void SdpClientL2CapCallback
                                              //       (uint16_t channelID,
  // ...                                      //        struct remdev *remdev)
};
// }}}
```

- Un tableau statique de 7 `struct sdpServInfo` dans le segment `bss`.
- Overflow depuis `struct sdpServInfo[n].pkt_data`.
- `n` est le numéro de client SDP …
- … associé au champs `channel ID` L2CAP.

Étape 1 / Écriture en mémoire





Objectif :

- Modifier un pointeur pour pouvoir écriture n'importe où

Étape 1 / Corruption de l'espace mémoire

```
struct sdpServInfo[0] {
  /* 0x0000 */  void    * next;
  /* 0x0004 */  void    * prev;
  /* 0x0008 */  uint8_t * ptr_pkt_data;
      // ...
  /* 0x0088 */  uint8_t   pkt_header [5];
  /* 0x008D */  uint8_t   pkt_data   [507];
};
struct sdpServInfo[1] {
  /* 0x0288 */  void    * next;
  /* 0x028C */  void    * prev;
  /* 0x0290 */  uint8_t * ptr_pkt_data;
      // ...
  /* 0x0310 */  uint8_t   pkt_header [5];
  /* 0x0314 */  uint8_t   pkt_data   [507];
}; // size = 648 (0x288) bytes
// ...
```

Étape 1 / Corruption de l'espace mémoire

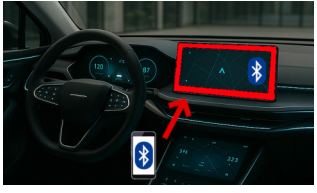
```
struct sdpServInfo[0] {
  /* 0x0000 */  void    * next;
  /* 0x0004 */  void    * prev;
  /* 0x0008 */  uint8_t * ptr_pkt_data;
      // ...
  /* 0x0088 */  uint8_t   pkt_header [5];
  /* 0x008D */  uint8_t   pkt_data   [507];
};
struct sdpServInfo[1] {
  /* 0x0288 */  void    * next;
  /* 0x028C */  void    * prev;
  /* 0x0290 */  uint8_t * ptr_pkt_data;
      // ...
  /* 0x0310 */  uint8_t   pkt_header [5];
  /* 0x0314 */  uint8_t   pkt_data   [507];
}; // size = 648 (0x288) bytes
// ...
```

Étape 1 / Corruption de l'espace mémoire

```
struct sdpServInfo[0] {
  /* 0x0000 */  void    * next;
  /* 0x0004 */  void    * prev;
  /* 0x0008 */  uint8_t * ptr_pkt_data;
      // ...
  /* 0x0088 */  uint8_t   pkt_header [5];
  /* 0x008D */  uint8_t   pkt_data   [507];
};
struct sdpServInfo[1] {
  /* 0x0288 */  void    * next;
  /* 0x028C */  void    * prev;
  /* 0x0290 */  uint8_t * ptr_pkt_data;
      // ...
  /* 0x0310 */  uint8_t   pkt_header [5];
  /* 0x0314 */  uint8_t   pkt_data   [507];
}; // size = 648 (0x288) bytes
// ...
```

Notre scénario : (avec 2 clients SDP)

1. Déclenchement d'un overflow depuis `struct sdpServInfo[0].pkt_data` *Client 1*
   - *MTU* = 8 => buffer de réponse SDP de taille max 65 535 octets.
   - Taille max plafonnée par rapport au type de services demandés :
     - L2CAP : 729 octets
     - Public Browse Group : 957 octets.

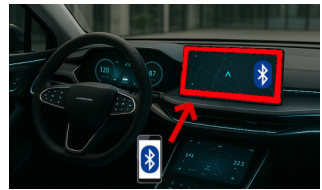


Étape 1 / Corruption de l'espace mémoire

```
struct sdpServInfo[0] {
  /* 0x0000 */  void    * next;
  /* 0x0004 */  void    * prev;
  /* 0x0008 */  uint8_t * ptr_pkt_data;
      // ...
  /* 0x0088 */  uint8_t   pkt_header [5];
  /* 0x008D */  uint8_t   pkt_data   [507];
};
struct sdpServInfo[1] {
  /* 0x0288 */  void    * next;
  /* 0x028C */  void    * prev;
  /* 0x0290 */  uint8_t * ptr_pkt_data;
      // ...
  /* 0x0310 */  uint8_t   pkt_header [5];
  /* 0x0314 */  uint8_t   pkt_data   [507];
}; // size = 648 (0x288) bytes
// ...
```

Notre scénario : (avec 2 clients SDP)

1. Déclenchement d'un overflow depuis
   `struct sdpServInfo[0].pkt_data` *Client 1*
2. Modification de l'adresse de
   `struct sdpServInfo[1].ptr_pkt_data` *Client 1*

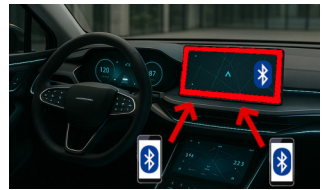


Étape 1 / Corruption de l'espace mémoire

```
struct sdpServInfo[0] {
  /* 0x0000 */  void    * next;
  /* 0x0004 */  void    * prev;
  /* 0x0008 */  uint8_t * ptr_pkt_data;
      // ...
  /* 0x0088 */  uint8_t   pkt_header [5];
  /* 0x008D */  uint8_t   pkt_data   [507];
};
struct sdpServInfo[1] {
  /* 0x0288 */  void    * next;
  /* 0x028C */  void    * prev;
  /* 0x0290 */  uint8_t * ptr_pkt_data;
      // ...
  /* 0x0310 */  uint8_t   pkt_header [5];
  /* 0x0314 */  uint8_t   pkt_data   [507];
}; // size = 648 (0x288) bytes
// ...
```

Notre scénario : (avec 2 clients SDP)

1. Déclenchement d'un overflow depuis
   `struct sdpServInfo[0].pkt_data` *Client 1*
2. Modification de l'adresse de
   `struct sdpServInfo[1].ptr_pkt_data` *Client 1*
3. Écriture à l'adresse pointée par
   `struct sdpServInfo[1].ptr_pkt_data` *Client 2*

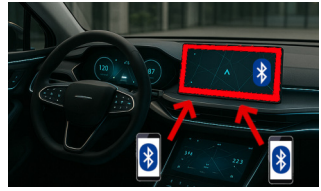


Étape 1 / Corruption de l'espace mémoire

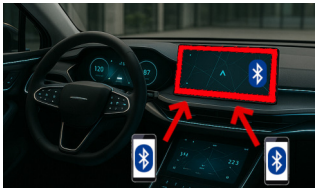
```
struct sdpServInfo[0] {
  /* 0x0000 */  void    * next;
  /* 0x0004 */  void    * prev;
  /* 0x0008 */  uint8_t * ptr_pkt_data;
      // ...
  /* 0x0088 */  uint8_t   pkt_header [5];
  /* 0x008D */  uint8_t   pkt_data   [507];
};
struct sdpServInfo[1] {
  /* 0x0288 */  void    * next; <= valeur quelconque
  /* 0x028C */  void    * prev; <= valeur quelconque
  /* 0x0290 */  uint8_t * ptr_pkt_data; <= &Adresse cible
      // ...
  /* 0x0310 */  uint8_t   pkt_header [5];
  /* 0x0314 */  uint8_t   pkt_data   [507];
}; // size = 648 (0x288) bytes
// ...
```

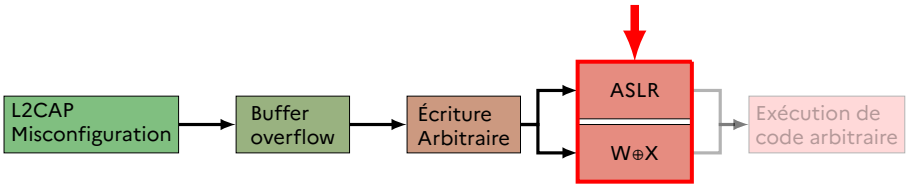
Notre scénario : (avec 2 clients SDP)

1. Déclenchement d'un overflow depuis
   `struct sdpServInfo[0].pkt_data` *Client 1*
2. Modification de l'adresse de
   `struct sdpServInfo[1].ptr_pkt_data` *Client 1*
3. Écriture à l'adresse pointée par
   `struct sdpServInfo[1].ptr_pkt_data` *Client 2*



Étape 2 / Contournement protections de sécurité





Objectifs :

- Obtenir une fuite d'information
- Rendre une page mémoire RW**X**

Étape 2 / Contournement protections de sécurité

- Via l'**étape 1** : **on ne dispose que d'une primitive d'écriture !**
  - débit très faible du canal d'écriture !

Étape 2 / Contournement protections de sécurité

- Via l'**étape 1** : **on ne dispose que d'une primitive d'écriture !**
  - débit très faible du canal d'écriture !

- ***Présence*** *d'****ASLR*** *(Address Space Layout Randomization)* :
  - obligation de trouver une fuite d'information
  - conception d'un script contournant l'implémentation de l'ASLR de la cible
- ***Présence*** *de* ***W⊕X*** *(Data Execution Prevention – DEP)* :
  - obligation de réutiliser le code existant
  - utilisation de gadgets (**étape 3**) (pour appeler **mprotect()**)
  - impossibilité de faire du ROP *(Return Oriented Programming)*
    - plusieurs threads
    - pile difficile à contrôler (on a un *heap overflow*)
    - écriture octet par octet

Étape 2 / Bypass de l'ASLR en détails

arch/arm/*

```
unsigned long arch_randomize_brk(struct mm_struct *mm)
{
   unsigned long range_end = mm->brk + 0x02000000;
   // Adresse alignée sur un frontière de PAGE
   return randomize_range(mm->brk, range_end, 0) ? : mm->brk;
}
```

Étape 2 / Bypass de l'ASLR en détails

arch/arm/*

```c
unsigned long arch_randomize_brk(struct mm_struct *mm)
{
   unsigned long range_end = mm->brk + 0x02000000;
   // Adresse alignée sur un frontière de PAGE
   return randomize_range(mm->brk, range_end, 0) ? : mm->brk;
}
```

- random_int : 0x02**XX_XXXX**

Étape 2 / Bypass de l'ASLR en détails

arch/arm/*

```c
unsigned long arch_randomize_brk(struct mm_struct *mm)
{
   unsigned long range_end = mm->brk + 0x02000000;
   // Adresse alignée sur un frontière de PAGE
   return randomize_range(mm->brk, range_end, 0) ? : mm->brk;
}
```

- random_int : 0x02**XX_XXXX**
- PAGE_ALIGN : 0x02**XX_X**000

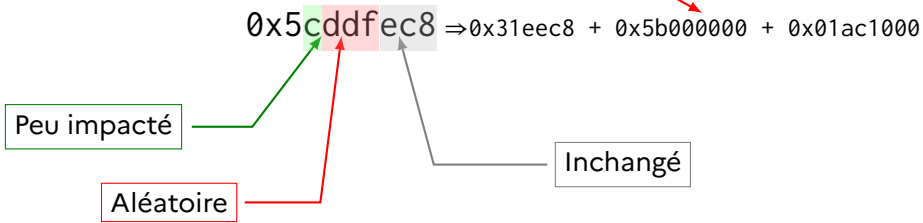
Étape 2 / Bypass de l'ASLR en détails

arch/arm/*

```
unsigned long arch_randomize_brk(struct mm_struct *mm)
{
    unsigned long range_end = mm->brk + 0x02000000;
    // Adresse alignée sur un frontière de PAGE
    return randomize_range(mm->brk, range_end, 0) ? : mm->brk;
}
```

- random_int : 0x02**XX_XXXX**
- PAGE_ALIGN : 0x02**XX_X**000

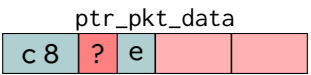
Exemple :



Étape 2 / Bypass de l'ASLR en détails

- En force brute, seulement `0x2000` = 8 192 possibilités !
- La primitive d'écriture permet d'écrire en dépassement de tampon
- Le `ptr_pkt_data`, est réécrit en commençant par ses octets de poids faibles cause : little-endian

Exemple : `0x31` `e` `ec8`



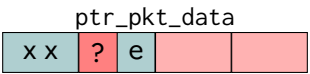
Conséquences

- On peut **réécrire de manière fiable 256 octets**
- On peut étendre la fenêtre de tir 16 536 octets en brute force ! *(seulement 16 possibilités à tester)*
  - Définition d'un oracle (utilisation du pointeur en `0x31ff88`)
- On peut alors obtenir les octets restants directement.

Étape 2 / Bypass de l'ASLR en détails

- En force brute, seulement `0x2000` = 8 192 possibilités !
- La primitive d'écriture permet d'écrire en dépassement de tampon
- Le `ptr_pkt_data`, est réécrit en commençant par ses octets de poids faibles cause : little-endian

Exemple : `0x31` `e` `ec8` ⇒ `0x31eexx`



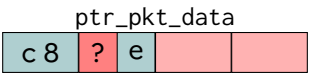
Conséquences

- On peut **réécrire de manière fiable 256 octets**
- On peut étendre la fenêtre de tir 16 536 octets en brute force ! *(seulement 16 possibilités à tester)*
  - Définition d'un oracle (utilisation du pointeur en `0x31ff88`)
- On peut alors obtenir les octets restants directement.

Étape 2 / Bypass de l'ASLR en détails

- En force brute, seulement `0x2000` = 8 192 possibilités !
- La primitive d'écriture permet d'écrire en dépassement de tampon
- Le `ptr_pkt_data`, est réécrit en commençant par ses octets de poids faibles cause : little-endian

Exemple : `0x31` `e` `ec8` ⇒ `0x31ff88`



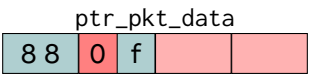
Conséquences

- On peut **réécrire de manière fiable 256 octets**
- On peut étendre la fenêtre de tir 16 536 octets en brute force ! *(seulement 16 possibilités à tester)*
  - Définition d'un oracle (utilisation du pointeur en `0x31ff88`)
- On peut alors obtenir les octets restants directement.

Étape 2 / Bypass de l'ASLR en détails

- En force brute, seulement `0x2000` = 8 192 possibilités !
- La primitive d'écriture permet d'écrire en dépassement de tampon
- Le `ptr_pkt_data`, est réécrit en commençant par ses octets de poids faibles cause : little-endian

Exemple : `0x31` `e` `ec8` ⇒ `0x31ff88`



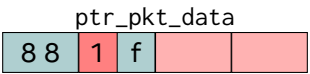
Conséquences

- On peut **réécrire de manière fiable 256 octets**
- On peut étendre la fenêtre de tir 16 536 octets en brute force ! *(seulement 16 possibilités à tester)*
  - Définition d'un oracle (utilisation du pointeur en `0x31ff88`)
- On peut alors obtenir les octets restants directement.

Étape 2 / Bypass de l'ASLR en détails

- En force brute, seulement `0x2000` = 8 192 possibilités !
- La primitive d'écriture permet d'écrire en dépassement de tampon
- Le `ptr_pkt_data`, est réécrit en commençant par ses octets de poids faibles cause : little-endian

Exemple : `0x31` `e` `ec8` ⇒ `0x31ff88`



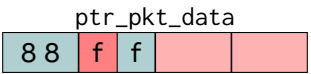
Conséquences

- On peut **réécrire de manière fiable 256 octets**
- On peut étendre la fenêtre de tir 16 536 octets en brute force ! *(seulement 16 possibilités à tester)*
  - Définition d'un oracle (utilisation du pointeur en `0x31ff88`)
- On peut alors obtenir les octets restants directement.

Étape 2 / Bypass de l'ASLR en détails

- En force brute, seulement `0x2000` = 8 192 possibilités !
- La primitive d'écriture permet d'écrire en dépassement de tampon
- Le `ptr_pkt_data`, est réécrit en commençant par ses octets de poids faibles cause : little-endian

Exemple : `0x31` `e` `ec8` ⇒ `0x31ff88`



Conséquences

- On peut **réécrire de manière fiable 256 octets**
- On peut étendre la fenêtre de tir 16 536 octets en brute force ! *(seulement 16 possibilités à tester)*
  - Définition d'un oracle (utilisation du pointeur en `0x31ff88`)
- On peut alors obtenir les octets restants directement.

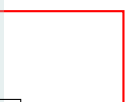
Étape 2 / Bypass de l'ASLR en détails

```
struct sdpServInfo[0] {
  /* 0x0000 */  void    * next;
  /* 0x0004 */  void    * prev;
  /* 0x0008 */  uint8_t * ptr_pkt_data;
      // ...
  /* 0x0060 */  struct sdpAttribInfo *sdpAttrib;
  /* 0x0064 */  struct sdpRecordInfo *sdpRecords;
      // ...
  /* 0x0088 */  uint8_t   pkt_header [5];
  /* 0x008D */  uint8_t   pkt_data   [507];
};
struct sdpServInfo[1] {
  /* 0x0288 */  void    * next;
  /* 0x028C */  void    * prev;
  /* 0x0290 */  uint8_t * ptr_pkt_data;
      // ...
  /* 0x0310 */  uint8_t   pkt_header [5];
  /* 0x0314 */  uint8_t   pkt_data   [507];
}; // size = 648 (0x288) bytes
// ...
```

Scénario :

Étape 2 / Bypass de l'ASLR en détails

```
struct sdpServInfo[0] {
  /* 0x0000 */  void    * next;
  /* 0x0004 */  void    * prev;
  /* 0x0008 */  uint8_t * ptr_pkt_data;
      // ...
  /* 0x0060 */  struct sdpAttribInfo *sdpAttrib;
  /* 0x0064 */  struct sdpRecordInfo *sdpRecords;
      // ...
  /* 0x0088 */  uint8_t   pkt_header [5];
  /* 0x008D */  uint8_t   pkt_data   [507];
};
struct sdpServInfo[1] {
  /* 0x0288 */  void    * next;
  /* 0x028C */  void    * prev;
  /* 0x0290 */  uint8_t * ptr_pkt_data;
      // ...
  /* 0x0310 */  uint8_t   pkt_header [5];
  /* 0x0314 */  uint8_t   pkt_data   [507];
}; // size = 648 (0x288) bytes
// ...
```



Scénario :

1. Déclenchement d'un overflow depuis `struct sdpServInfo[0].pkt_data`
2. `struct sdpServInfo[1].ptr_pkt_data` pointe *à l'intérieur* de `struct sdpServInfo[0].pkt_data`

Étape 2 / Bypass de l'ASLR en détails

```
struct sdpServInfo[0] {
  /* 0x0000 */  void    * next;
  /* 0x0004 */  void    * prev;
  /* 0x0008 */  uint8_t * ptr_pkt_data;
      // ...
  /* 0x0060 */  struct sdpAttribInfo *sdpAttrib;
  /* 0x0064 */  struct sdpRecordInfo *sdpRecords;
      // ...
  /* 0x0088 */  uint8_t   pkt_header [5];
  /* 0x008D */  uint8_t   pkt_data   [507];
};
struct sdpServInfo[1] {
  /* 0x0288 */  void    * next;
  /* 0x028C */  void    * prev;
  /* 0x0290 */  uint8_t * ptr_pkt_data;
      // ...
  /* 0x0310 */  uint8_t   pkt_header [5];
  /* 0x0314 */  uint8_t   pkt_data   [507];
}; // size = 648 (0x288) bytes
// ...
```

Scénario :

1. Déclenchement d'un overflow depuis `struct sdpServInfo[0].pkt_data`
2. `struct sdpServInfo[1].ptr_pkt_data` pointe *à l'intérieur* de `struct sdpServInfo[0].pkt_data`
3. le Client 2 fait une requête legitime et déclenche la fuite d'information.

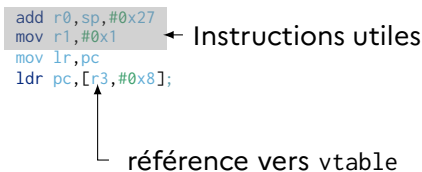
Étape 2 / Utilisation de JOP

- Utilisation classique avec un dispatcher gadget
  - pas facilement disponible dans l'implémentation
- `libbluego` est une bibliothèque **orientée object** en C++
  - impliquant des structures de type `vtables`
  - ex. de transcription de l'appel à une méthode virtuelle au sein de `libbluego`

```
add r0,sp,#0x27
mov r1,#0x1
mov lr,pc
ldr pc,[r3,#0x8];
```

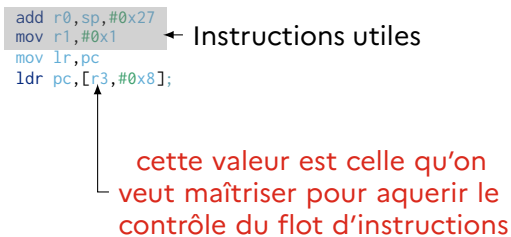
Étape 2 / Utilisation de JOP

- Utilisation classique avec un dispatcher gadget
  - pas facilement disponible dans l'implémentation
- `libbluego` est une bibliothèque **orientée object** en C++
  - impliquant des structures de type `vtables`
  - ex. de transcription de l'appel à une méthode virtuelle au sein de `libbluego`

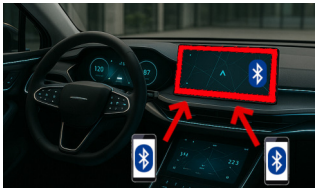


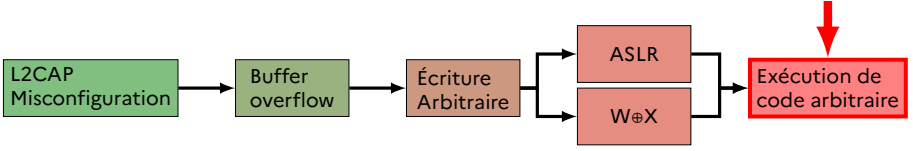
Étape 2 / Utilisation de JOP

- Utilisation classique avec un dispatcher gadget
  - pas facilement disponible dans l'implémentation
- `libbluego` est une bibliothèque **orientée object** en C++
  - impliquant des structures de type `vtables`
  - ex. de transcription de l'appel à une méthode virtuelle au sein de `libbluego`



Étape 3 / Exécution de code arbitraire

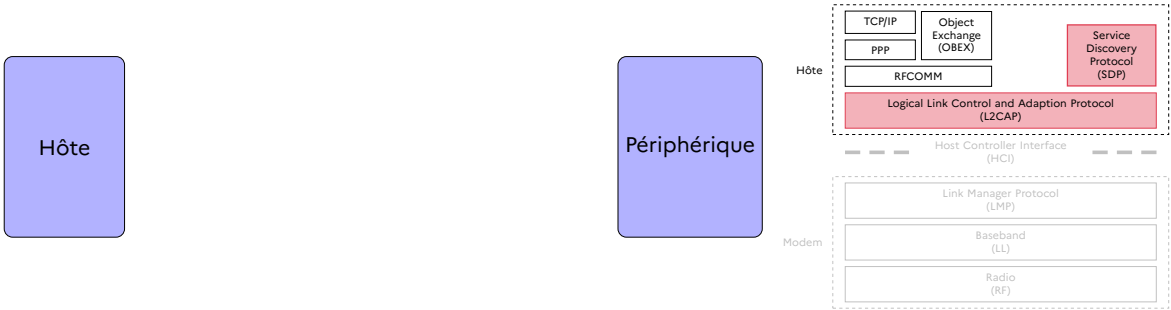




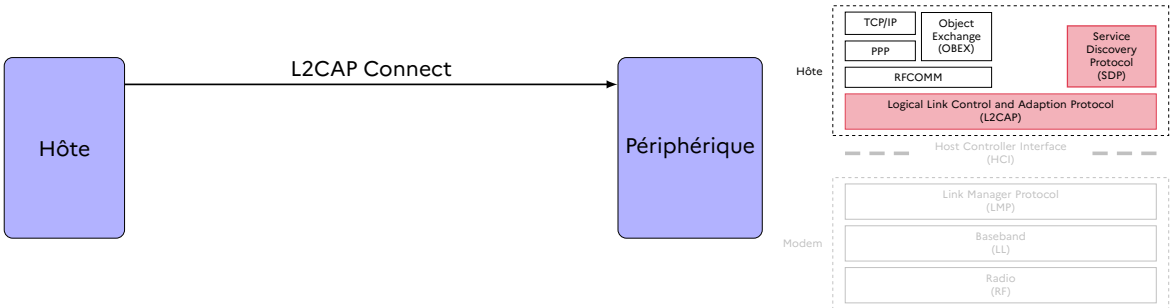
Objectifs :

- Trouver un pointeur de fonction (*callback*)
- Trouver un moyen de déclencher cette *callback* sans intervention de l'utilisateur
- Laisser le système opérationnel

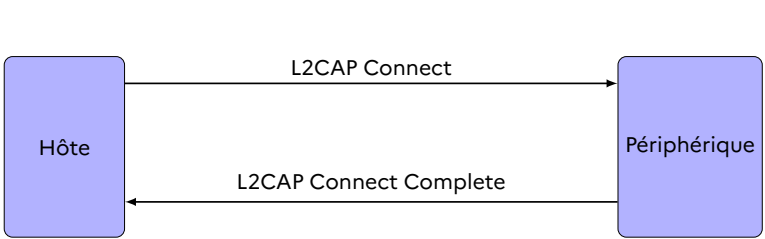
Étape 3 / Gestion de canal non sécurisé *(SDP uniquement)*

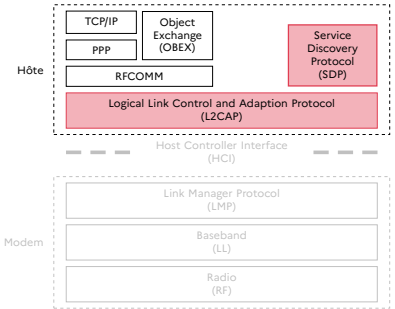


Étape 3 / Gestion de canal non sécurisé *(SDP uniquement)*

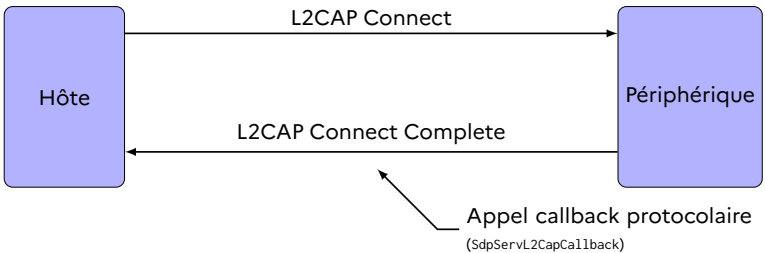


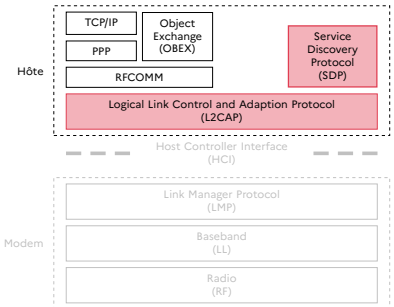
Étape 3 / Gestion de canal non sécurisé *(SDP uniquement)*



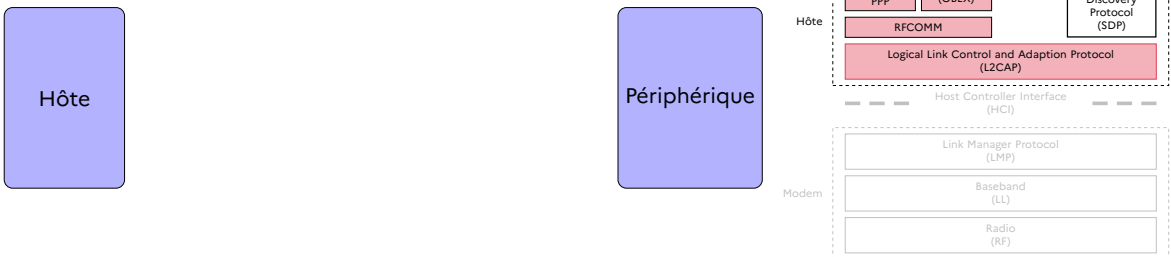


Étape 3 / Gestion de canal non sécurisé *(SDP uniquement)*

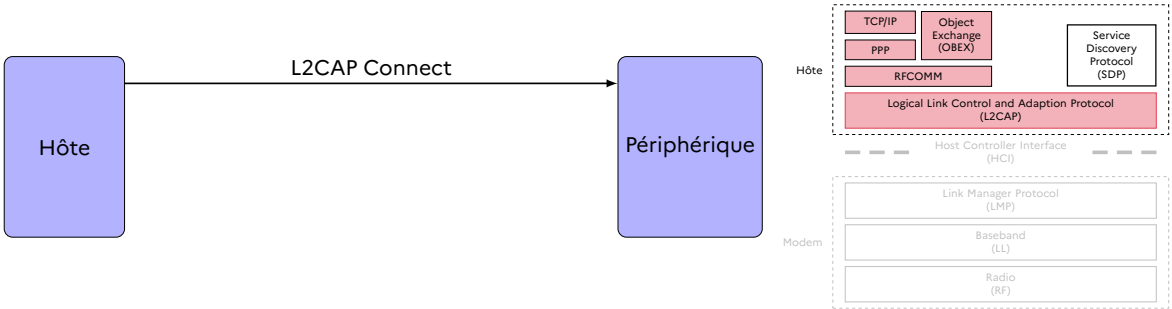




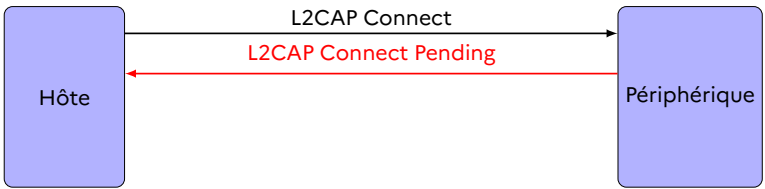
Étape 3 / Gestion de canal sécurisé

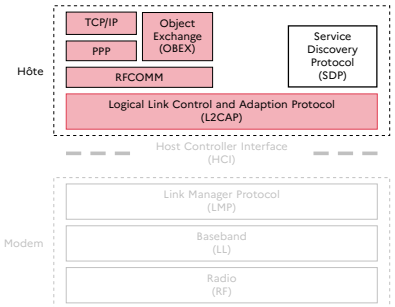


Étape 3 / Gestion de canal sécurisé

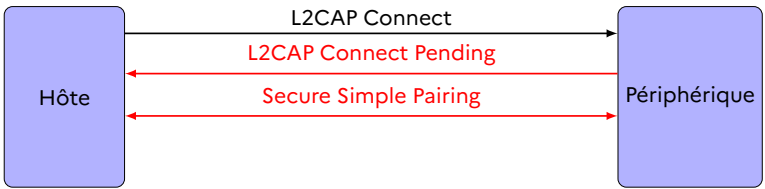


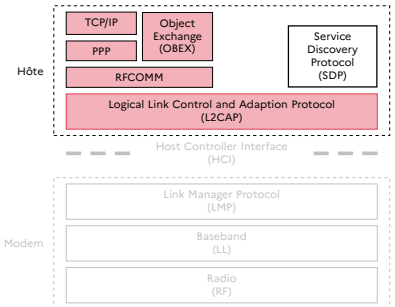
Étape 3 / Gestion de canal sécurisé



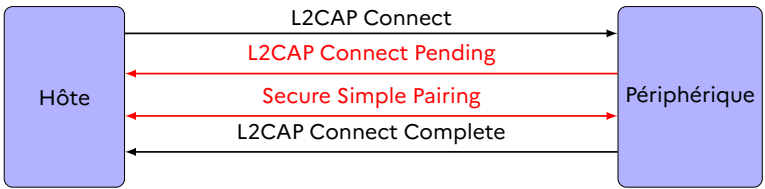


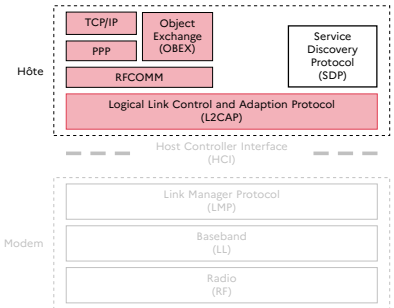
Étape 3 / Gestion de canal sécurisé



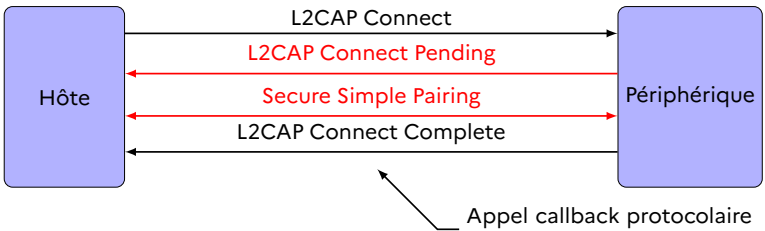


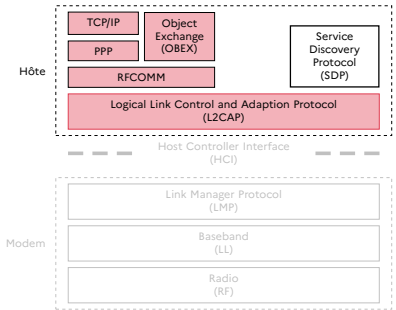
Étape 3 / Gestion de canal sécurisé



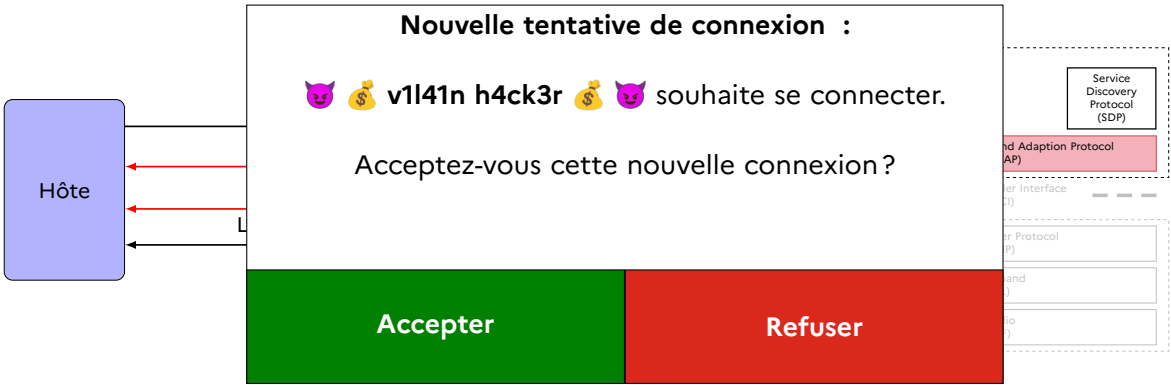


Étape 3 / Gestion de canal sécurisé

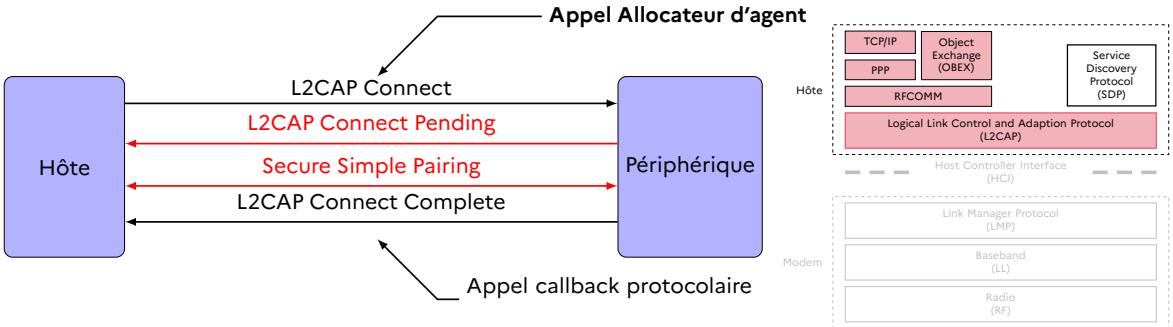




Étape 3 / Gestion de canal sécurisé



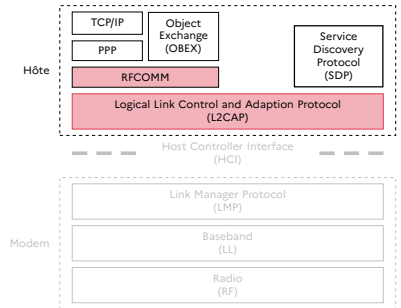
Étape 3 / Gestion de canal sécurisé



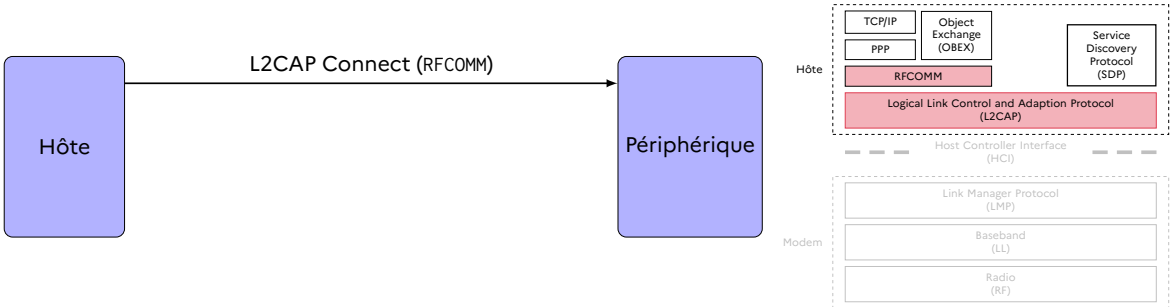
Étape 3 / Déclenchement de l'exécution

Hôte

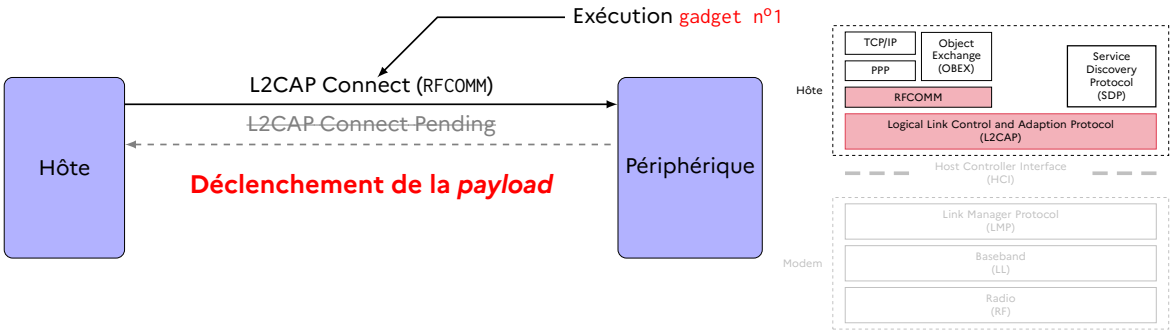
Périphérique



Étape 3 / Déclenchement de l'exécution



Étape 3 / Déclenchement de l'exécution



Impact de sécurité

Conséquences de la vulnérabilité

- Execution de code arbitraire sur l'infodivertissement
  - dans un processus ayant des droits système.
- Possibilité d'exécuter des commandes CAN depuis le code arbitraire
  - **exécution à distance de commandes CAN**

Impact de sécurité

Conséquences de la vulnérabilité

- Execution de code arbitraire sur l'infodivertissement
  - dans un processus ayant des droits système.
- Possibilité d'exécuter des commandes CAN depuis le code arbitraire
  - **exécution à distance de commandes CAN**

Impact d'autres véhicules de modèles et de marques différences non mise à jour

Impact de sécurité

Conséquences de la vulnérabilité

- Execution de code arbitraire sur l'infodivertissement
  - dans un processus ayant des droits système.
- Possibilité d'exécuter des commandes CAN depuis le code arbitraire
  - **exécution à distance de commandes CAN**

Impact d'autres véhicules de modèles et de marques différences non mise à jour

Divulgation responsable

- Travail réalisé entre septembre 2023 et avril 2024.
- Le constructeur a été informé en **septembre 2024**.
- Réception positive et réaction très rapide de sa part.
- Le constructeur précise que les commandes CAN critiques sont filtrées.

Cette vulnérabilité a été corrigé par le constructeur du véhicule cible.

Conclusion

- Analyse d'une pile Bluetooth embarquée :
  - véhicule représentatif des années 2020 et **encore en circulation**.
- Présence d'une vulnérabilité référencée par la `CVE-2018-20378` :
  - initialement considérée comme non exploitable dans la version embarquée de la bibliothèque.
  - les détails publics d'exploitation ne sont pas applicables en l'état.
- Analyse menée en **boîte noire**.
- Proposition d'un scénario d'exploitation original :
  - attaque **à distance** et **sans interaction utilisateur**.

Conclusion

- Analyse d'une pile Bluetooth embarquée :
  - véhicule représentatif des années 2020 et **encore en circulation**.
- Présence d'une vulnérabilité référencée par la `CVE-2018-20378` :
  - initialement considérée comme non exploitable dans la version embarquée de la bibliothèque.
  - les détails publics d'exploitation ne sont pas applicables en l'état.
- Analyse menée en **boîte noire**.
- Proposition d'un scénario d'exploitation original :
  - attaque **à distance** et **sans interaction utilisateur**.
- Correctif déployé par le constructeur !

Conclusion

- Analyse d'une pile Bluetooth embarquée :
  - véhicule représentatif des années 2020 et **encore en circulation**.
- Présence d'une vulnérabilité référencée par la `CVE-2018-20378` :
  - initialement considérée comme non exploitable dans la version embarquée de la bibliothèque.
  - les détails publics d'exploitation ne sont pas applicables en l'état.
- Analyse menée en **boîte noire**.
- Proposition d'un scénario d'exploitation original :
  - attaque **à distance** et **sans interaction utilisateur**.
- Correctif déployé par le constructeur !

La sécurité des véhicules a un impact direct sur leur sûreté de fonctionnement.

Conclusion

- Analyse d'une pile Bluetooth embarquée :
  - véhicule représentatif des années 2020 et **encore en circulation**.
- Présence d'une vulnérabilité référencée par la `CVE-2018-20378` :
  - initialement considérée comme non exploitable dans la version embarquée de la bibliothèque.
  - les détails publics d'exploitation ne sont pas applicables en l'état.
- Analyse menée en **boîte noire**.
- Proposition d'un scénario d'exploitation original :
  - attaque **à distance** et **sans interaction utilisateur**.
- Correctif déployé par le constructeur !

La sécurité des véhicules a un impact direct sur leur sûreté de fonctionnement.

La surface d'attaque continue de s'élargir :

- Véhicules électriques : interfaces *Vehicle-to-Charger* (V2C) (ex. SSTIC 2019).
- Communication *Vehicle-to-Vehicle* (V2V), encore peu déployée, mais déjà critique.

Conclusion

- Analyse d'une pile Bluetooth embarquée :
  - véhicule représentatif des années 2020 et **encore en circulation**.
- Présence d'une vulnérabilité référencée par la `CVE-2018-20378` :
  - initialement considérée comme non exploitable dans la version embarquée de la bibliothèque.
  - les détails publics d'exploitation ne sont pas applicables en l'état.

[Beaucoup] plus de détails dans l'article SSTIC 2025 associé !

https://www.sstic.org/2025/presentation/300_secondes_chrono__prise_de_contrle_dun_infodivertissement_automobile__distance/

La sécurité des véhicules a un impact direct sur leur sûreté de fonctionnement.

La surface d'attaque continue de s'élargir :

- Véhicules électriques : interfaces *Vehicle-to-Charger* (V2C) (ex. SSTIC 2019).
- Communication *Vehicle-to-Vehicle* (V2V), encore peu déployée, mais déjà critique.

Questions ?

Philippe Trébuchet & Guillaume Bouffard
Laboratoire Architectures Matérielles et logicielles
Tour Mercure
31, quai de Grenelle
75015 Paris

1. Introduction
   - La sécurité des véhicules connectées
2. Analyse de sécurité du Firmware
3. MTU et overflows
   - Étape 1 : Exploitation du Buffer overflow
   - Étape 2 : Contournement mécanismes de sécurité
   - Étape 3 : Exécution de code arbitraire
4. Conclusion et Perspectives