Análise forense de sistemas de infoentretenimento em veículos BMW

Mestrado em Cibersegurança e Informática Forense

Ricardo Manuel da Costa Marques

Leiria, abril de 2023

Análise forense de sistemas de infoentretenimento em veículos BMW

Mestrado em Cibersegurança e Informática Forense

Ricardo Manuel da Costa Marques

Projeto de Mestrado realizado sob a orientação do Professor Doutor Miguel Cerdeira Marreiros Negrão, Professor da Escola Superior de Tecnologia e Gestão do Instituo Politécnico de Leiria.

Leiria, abril de 2023

Originalidade e Direitos de Autor

O presente relatório de projeto é original, elaborado unicamente para este fim, tendo sido devidamente citados todos os autores cujos estudos e publicações contribuíram para o elaborar.

Reproduções parciais deste documento serão autorizadas na condição de que seja mencionado o Autor e feita referência ao ciclo de estudos no âmbito do qual o mesmo foi realizado, a saber, Curso de Mestrado em Cibersegurança e Informática Forense, no ano letivo 2022/2023 da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, Portugal, e, bem assim, à data das provas públicas que visaram a avaliação destes trabalhos.

Dedicatória

Dedico este trabalho à minha esposa e aos meus filhos, por todo o apoio prestado nesta fase. Foi de facto um período muitíssimo difícil de conseguir conciliar, em termos de aulas, o trabalho e o apoio à família.

Agradecimentos

A conclusão do mestrado em Cibersegurança e Informática Forense, só foi possível com ajuda e apoio da minha família, nomeadamente minha esposa e os meus filhos, por serem a minha fonte de inspiração e motivação, que estiveram presentes a nível pessoal e académico durante todo este percurso.

Gostaria de agradecer ao meu orientador, Professor Doutor Miguel Cerdeira Marreiros Negrão do Instituto Politécnico de Leiria, por todo o apoio, paciência, orientação e disponibilidade que teve ao longo do processo do desenvolvimento deste projeto.

Agradeço ainda a oportunidade que este projeto me deu em trabalhar e explorar a área de análise forense digital de sistemas in-vehicle infotainment (IVI) dos veículos da marca BMW.

Resumo

A Análise forense digital emprega várias técnicas, aplicações e procedimentos para a recolha, preservação e análise de provas digitais em vários dispositivos. Atualmente as aplicações na análise forense digital são imensas e vão desde a análise de um disco rígido até à análise de um smartphone, mas existem outras fontes de provas com muito potencial que são os sistemas in-vehicle infotainment (IVI) presentes nos veículos. A Análise aos sistemas IVI é importante, permite encontrar vários dados importantes para investigar e esclarecer determinados crimes e acidentes de viação. Para efeitos deste projeto optou-se por focar numa das áreas da análise forense digital, os sistemas IVI dos veículos, especificamente os da marca BMW.

Estes sistemas IVI possuem muitos dados importantes, como dados de contatos, histórico de chamadas, mensagens, e-mail, dados do Bluetooth, modelos e marca de telefone, dados de navegação GPS, músicas, vídeos e ainda permitem interação do telefone com os sistemas IVI através de aplicações Android Auto e Apple Carplay.

Neste sentido, foram analisados vários sistemas IVI da marca BMW. Foi analisado o sistema de ficheiros desse mesmo sistema e quais os dados contidos nas partições que o compõem. Foi analisado as bases de dados SQLite presente nos sistemas IVI BMW, foi estudado quais os dados continham e se eram úteis para analise forense digital. Foram desenvolvidos dois plugins ingest module para a ferramenta Autopsy, para analisar as bases de dados SQLite, que contêm os dados mais importantes para o investigador. Os módulos desenvolvidos permitem recuperar vários dados relevantes nas diversas bases de dados, como históricos de chamadas, contatos, mensagens, histórico da web, dados do Bluetooth, marca e modelo do telefone, gostos musicais e quais os dispositivos foram ligados aos sistemas IVI da BMW.

Palavras-chave: Vehicle, Infotainment, forense, análise, prova, BMW.

Abstract

Digital Forensics employs various techniques, applications and procedures for the collection, preservation and analysis of digital evidence on various devices. Currently the applications in digital forensic analysis are immense and range from the analysis of a hard disk to the analysis of a smartphone, but there are other sources of evidence with great potential that are the in-vehicle infotainment systems (IVI) present in vehicles. The analysis of the IVI systems is important, it allows finding several important data to investigate and clarify certain crimes and traffic accidents. For the purposes of this project, it was decided to focus on one of the areas of digital forensic analysis, the IVI systems of vehicles, specifically those of the BMW brand.

These IVI systems have many important data, such as contact data, call history, messages, e-mail, Bluetooth data, phone models and brand, GPS navigation data, music, videos and even allow interaction between the phone and the systems. IVI through Android Auto and Apple Carplay applications.

In this sense, several IVI systems from the BMW brand were analyzed. The file system of that same system was analyzed and the data contained in the partitions that compose it. The SQLite databases present in the BMW IVI systems were analyzed, it was studied what data they contained and whether they were useful for digital forensic analysis. Two ingest module plugins were developed for the Autopsy tool, to analyze SQLite databases, which contain the most important data for the researcher. The developed modules make it possible to recover various relevant data in the various databases, such as call histories, contacts, messages, web history, Bluetooth data, phone brand and model, musical tastes and which devices were connected to BMW's IVI systems.

Keywords: Vehicle, infotainment, forensics, analysis, evidence, BMW.

Índice

Lista de Figuras

Lista de tabelas

Lista de siglas e acrónimos

AES	Advanced Encryption Standard
AFF	Advanced Forensic Format
BSP	Board Support Package
CAN	Controller Area Network
CCC	Car Communication Computer
CCTV	Closed Circuit Television
CDR	Crash Data Retrieval
CIC	Computer-In-Car
CID	Central Information Display
CPU	Central Process Unit
ECU	Electronic Control Unit or Engine
EDR	Event Data Recorder
FAC	Final Assembly Code
FTK	Forensics Toolkit
GPS	Global Positioning System
GSM	Global System for Mobile
HDD	Hard Disk Drive
HMI	Human–Machine Interface
IDE	Integrated Development Environment
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
IOT	Internet of Things
IP	Internet Protocol

ISP	In System Programming
IVI	in-vehicle infotainment
JTAG	Join Test Action Group
JVM	Java Virtual Machine
MCC	Mobile Country Code
MNC	Mobile Network Code
NAD	Node Address for Diagnostics
NBT	Next Big Thing
OBD	On-Board Diagnostic
RAM	Random Access Memory
RTOS	Neutrino Realtime Operation System
SIM	Subscriber Identity Module
SMS	Short Message Service
SSD	Solid State Drives
TAC	Type Allocation Code
TAP	Test Access Port
TMC	Traffic Message Channel
TSK	The Sleuth Kit
UI	User Interface
USB	Universal Serial Bus
VIN	vehicle identification number
VM	virtual machine

1. Introdução

No âmbito do 2.º ano do mestrado em Cibersegurança e Informática Forense da Escola Superior de Tecnologia e Gestão de Leira (ESTG) do Instituto Politécnico de Leiria, foi elaborado este projeto sobre "Analise forense de sistemas de infoentretenimento em veículos BMW". Este tema está inserido na área científica de Análise Forense Digital.

O aspeto mais importante numa análise forense digital é a extração de dados para serem apresentados como prova [1]. A prova é a informação guardada ou transmitida com valor probatório, que pode demonstrar factos relevantes, com elementos que estão relacionados com a preparação e a prática do crime.

Na área de análise forense digital, o princípio de Locard e a informática unem-se. O princípio de troca de Locard diz que, no mundo atual, quando os autores entram ou abandonam um cenário de crime, deixam algum vestígio e levam algo com eles. Os exemplos forenses antigos incluíam DNA, impressões digitais, cabelo, entre outros. Na ciência de análise forense digital existe também um conjunto de vestígios e registos que podem ser deixados para trás, como por exemplo: ficheiros de registos dos sistemas, dados e outras informações [2].

Uma das indústrias que tem conseguido introduzir novos equipamentos e dispositivos para os seus clientes é a indústria automóvel. Com a introdução de equipamentos de infoentretenimento denominados como in-vehicle Infotainment, permite o armazenamento de informações a partir de sincronizações com outros equipamentos, neste caso a sincronização de telefones [1]. Na sincronização com o telefone, existe uma grande troca de informação do telefone com o veículo, nomeadamente, contactos, históricos de chamadas, SMS, endereço Bluetooth, International Mobile Equipment Identity (IMEI), International Mobile Subscriber Identity (IMSI), modelo e marca do telefone.

Os veículos modernos transformaram-se em computadores com rodas, possuindo no seu interior armazenamento através de discos rígidos e cartões SD para guardar milhões de dados. Possuem um sistema operativo, memoria *RAM* e processadores. Estes sistemas possuem ligações Wi-Fi, Bluetooth, rádio, navegação Global Positioning System (GPS) e

por último permitem passar as aplicações do telefone para o veículo através do Android Auto e Apple CarPlay [3].

Os veículos são uma rede complexa de sistemas eletrónicos altamente integrados. De acordo com a Berla Vehicle Forensics, veículos típicos contêm mais de 75 sistemas de computador, executam mais de 150 milhões de linhas de código e geram mais de 25 GB de dados por hora [4].

Os sistemas IVI, estão presentes em todos os veículos modernos, tendo como objetivo fornecer informação e entretenimento aos condutores e aos seus passageiros.

A análise forense digital aos sistemas IVI fornece aos investigadores a capacidade de preservar uma ampla gama de provas digitais e será cada vez mais utilizada em investigações criminais na próxima década [5].

A investigação forense a sistemas IVI permite saber o histórico da navegação GPS, a data e hora dos locais onde passou, qual a localização, destinos recentes, locais guardados e rotas anteriores, permite obter o histórico de chamadas e mensagens de um telefone que tenha sincronizado com o veículo através de Bluetooth. Estes dados são importantes para saber o histórico de localização depois de um furto do veículo ou quando e onde ocorreu um acidente. As informações armazenadas nos sistemas IVI podem conter provas muito importantes para a investigação de acidentes de viação e criminais.

Previsivelmente, os veículos modernos são uma importante fonte de provas numa investigação, e a análise forense de sistemas IVI é uma área de pesquisa emergente.

Um sistema IVI pode ser definido como um sistema que comunica, permite a sincronização com o telefone via Bluetooth e Wi-Fi, para fornecer aos condutores e ocupantes informações especificas do veículo, navegação via GPS, aplicações integradas ou de entretenimento, incluindo áudio e vídeo, permitindo efetuar chamadas, mensagens, permitindo fornecer conteúdo para fins formativos e de entretenimento [3].

Para a sua recolha e análise de provas digitais usamos as melhores práticas mencionadas na Scientific Working Group on Digital Evidence (SWGDE) [6]. Para a análise do conteúdo obtido na aquisição é necessário recorrer às várias ferramentas de análise existentes. O método de aquisição varia de acordo com a marca e o modelo do veículo. Muitos sistemas de IVI enfrentam várias barreiras na fase de aquisição, sendo necessário aplicar algumas

técnicas forenses, nomeadamente a técnica de *chip-off* ou aplicar outras técnicas nomeadamente Join Test Action Group (JTAG) e In System Programming (ISP). Existem alguns sistemas IVI que possuem discos rígidos, neste caso torna-se mais fácil a sua análise.

Para realizar este projeto, no início foi necessário restringir a análise apenas a veículos que usassem como armazenamento de memória discos rígidos (HDD), por forma a facilitar a obtenção de dados para análise no contexto deste trabalho. Com base nesse critério foi escolhida a marca BMW, onde analisamos 4 modelos diferentes.

Neste projeto descrevemos como aceder e desmontar aos sistemas IVI da marca BMW, como remover o disco rígido do veículo, como analisar esses dados e quais os dados mais relevantes para os investigadores.

Para obter os dados foram usadas várias ferramentas open-source, nomeadamente o FTK Imager, para obter a imagem física dos discos rígidos dos sistemas IVI, e a ferramenta Autopsy para efetuar a sua análise.

Houve uma etapa de exploração dos ficheiros existentes nas imagens dos discos rígidos que foram analisados, e onde se determinou quais eram os ficheiros que continham informação relevante e qual o seu formato.

Para otimizar o processo foram desenvolvidos 2 plugins de tipo “Ingest Modules” para a ferramenta Autopsy, para permitir efetuar uma análise e encontrar os dados relevantes que se encontram nas bases de dados SQLite.

1.1.Motivação

Uma motivação importante para este trabalho é análise forense digital aos sistemas IVI da marca BMW e explorar os seus dados. Atualmente os veículos incorporam inúmeras tecnologias e contém grande quantidades de dados, permitindo aumentar a interatividade entre o veículo e o condutor.

É motivante analisar os sistemas IVI, devido à quantidade de dados que esses sistemas IVI possuem. Estes sistemas IVI possuem muitas bases de dados SQLite, os dados encontrados nos veículos permitem que possam ser analisados e ajudam os investigadores num cenário de crime ou acidente. Os veículos possuem vários dados, como contatos, registo de chamadas, *SMS*, dados de navegação GPS, mac address do Bluetooth, IMSI, IMEI, músicas. Logo todos estes dados são importantes, a extração desse conhecimento é muito valiosa.

A outra motivação é tornar mais rápida a análise forense de sistemas IVI de veículos da marca BMW, para isso é necessário desenvolver dois plugins ingest module para a ferramenta Autopsy analisar e encontrar as bases de dados SQlite de forma automatizada e rápida, permitindo auxiliar os investigadores e processar facilmente os dados relevantes.

1.2. Objetivos

Um dos objetivos deste projeto é a exploração e estudo do processo de aquisição de dados nos sistemas IVI em veículos da marca BMW, bem como a análise de dados obtidos na fase de aquisição. Consiste também em identificar que tipo de dados e informações são armazenadas nos sistemas IVI da marca BMW.

Este projeto também tem o objetivo de fornecer os procedimentos, praticas para aquisição, preservação e análise dos sistemas IVI.

A quantidade de dados que espero encontrar inclui, contatos, mensagens, registo de chamadas, quais os dispositivos que se ligaram ao veículo, mac address do Bluetooth, IMSI, IMEI, modelo, marca do telefone, músicas, dados de navegação GPS.

Este projeto consiste ainda no desenvolvimento de dois plugins “ingest modules” para a ferramenta forense Autopsy, que permitem efetuar uma análise forense aos dados adquiridos nas diversas bases de dados SQLite, permitindo automatizar o processo e apresentar ao investigador os dados mais importantes dessa mesma análise.

1.3. Contributos

O trabalho de investigação conducente à realização deste projeto, teve um duplo contributo, derivado dos objetivos inicialmente definidos. Este projeto efetua análise e exploração de dados dos sistemas IVI dos modelos CIC e NBT dos veículos de Marca BMW.

Este projeto permitiu criar dois plugins ingest module em Python para a ferramenta forense Autopsy, para automatizar, detetar e analisar as bases de dados dos sistemas IVI da BMW, e a partir daí identificar os dados mais importantes a nível forense.

1.4.Organização do documento

O presente documento encontra-se dividido em sete capítulos, e está organizado da seguinte forma. O presente capítulo (primeiro), apresenta o enquadramento do tema, com a introdução

ao mesmo, já o segundo capítulo, apresenta os conceitos da análise forense digital e os vários ramos da análise forense digital nos veículos, nomeadamente análise forense digital nos sistemas IVI.

O terceiro capítulo expõe as ferramentas e tecnologias que foram utilizadas durante a elaboração deste projeto, contempla ainda neste capítulo a ferramenta Berla, a única ferramenta paga existente no mercado que permite analisar os sistemas IVI.

O quarto capítulo, descreve o sistema e funcionalidades do sistema IVI da BMW, como se adquire e remove o sistema IVI do veículo, aquisição da imagem dos discos rígidos dos sistemas IVI, como se montam as partições, organização interna do sistema IVI e a localização e conteúdo das bases de dados.

O quinto capítulo, relata o desenvolvimento e o funcionamento dos dois plugins ingest modules para serem implementados na ferramenta Autopsy, que permite analisar de forma automatizada as diversas bases de dados SQLite dos sistemas IVI que foram analisados neste projeto.

O sexto capítulo, expõe a apresentação dos resultados dessas análises, com a comparação dos vários sistemas IVI que foram analisados. Por fim, o sétimo capítulo, contempla as principais conclusões deste projeto, sugerindo ainda trabalho futuro.

2. Análise Forense Digital em veículos

Neste capítulo será abordada de forma genérica o que é a análise forense digital, bem como alguns conceitos legais para dar ao leitor uma base adequada e maior compreensão das diferentes etapas durante o processo da análise forense digital. Existem várias formas de definir a análise forense digital, tendo cada autor a sua interpretação.

Segundo os autores Whatson e Jones Andrew [7] a análise forense digital, é um campo em crescimento que trata de recolher provas de dispositivos digitais, aplicando processos e procedimentos que possam conter provas. Este autor considera que os principais processos digitais forenses são, a preservação, identificação, extração, documentação, interpretação e a apresentação das provas obtidas.

Temos também uma definição descritiva de análise forense digital, que foi publicada pela Digital Forensics Research Conference em 2001 [8], que definiu a análise forense digital como o uso de métodos cientificamente comprovados para a preservação, recolha, validação, identificação, análise, interpretação, documentação e apresentação de provas digitais derivadas de fontes digitais com a finalidade de facilitar ou promover a reconstrução de eventos considerados criminosos, ou ajudar a antecipar ações não autorizadas que se mostrem prejudiciais às operações planeadas.

Segundo os Autores Mário Antunes e Baltazar Rodrigues, a análise forense digital pode definir-se como inspeção sistemática e técnica de um sistema informático e dos seus componentes (dados, hardware) para obtenção de provas digitais, ou evidências, de um crime ou qualquer outro uso ilícito que possa ser investigado. Durante a identificação e manuseio das evidências, deve-se garantir a integridade dos dados [9].

Tendo em conta a informação dos vários autores, conseguimos entender o conceito de análise forense digital, que tem como objetivo identificar, preservar, recuperar, analisar e apresentar os factos [10].

A análise forense digital tem como objetivo principal a recuperação e investigação de provas digitais, e pode ser dividida em vários ramos, como analise forense em veículos, computadores e smartphones, cartões de memória, disquetes, memórias flash, *RAM*, *IOT*, e-mail, redes, discos HDD/SSD, sistemas de navegação GPS, Smartwatchs, Smart Tv, Home

Kits, consolas de jogos, Drones, CCTV e PDAs. Na figura 1 estão identificadas diversas áreas de intervenção da atividade da Análise Forense Digital.

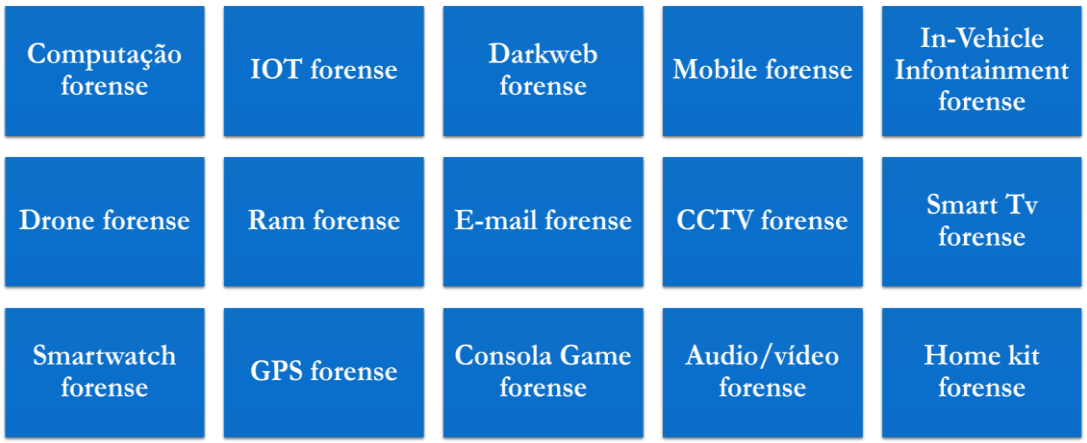


Figura 1 - Diversas áreas da Análise Forense Digital

A análise forense digital constitui uma peça fundamental no processo probatório, servindo para responder às perguntas típicas da investigação de um ilícito ou incidente: "quê"; "quem"; "quando"; "como"; e, em alguns casos, "onde" e "porquê".

O principal objetivo da análise forense digital é produzir uma reconstrução cientificamente embasada de eventos, auxiliando no processo da recolha de evidências que possam ditar a inocência ou culpa de um suspeito. Como ferramenta útil de apoio ao processo de tomada de decisão, a análise forense digital tem uma presença crescente no paradigma judicial português e no contexto organizacional [11].

Essas evidências digitais, precisam de respeitar alguns princípios fundamentais para que sejam legalmente admissíveis e aprovadas:

- Admissibilidade, a prova deve estar de acordo com a legislação em vigor.
- Autenticidade, deve ser autêntica na relação entre a pista e o incidente.
- Integridade, tem de ser completa, sem faltar nada.
- Confiabilidade, na descrição de todas as ações realizadas no tratamento de evidências precisa ser confiável.
- Credibilidade, a evidência deve ser compreensível e plausível.

Portanto, as provas digitais devem ser legalmente admissíveis e a forma como foram obtidas deve ser claramente conhecida. Também deve ser tecnicamente incontestável: a sua origem deve ser verificável e a sua integridade deve ser demonstrada.

Os investigadores entendem o valor das informações extraídas dos equipamentos e precisam respeitar o facto de que podem ser facilmente comprometidas se não forem devidamente manuseadas e protegidas. Por esse motivo, é fundamental estabelecer e seguir diretrizes e protocolos rígidos nas investigações de forma a preservar as provas digitais. Esses procedimentos incluem instruções detalhadas sobre o período em que os investigadores estão autorizados a recuperar possíveis provas digitais, como preparar adequadamente os sistemas para recuperação de provas, onde armazenar qualquer prova recuperada e como documentar essas atividades para ajudar a garantir a autenticidade dos dados.

A análise digital forense é efetuada com recurso a perícias, cujo resultado é um relatório técnico, onde são descritos detalhadamente os equipamentos analisados e os resultados obtidos [9].

2.1.Metodologia da Análise Forense Digital

Existem diferentes pontos de vista sobre as fases do processo de análise forense digital. A metodologia usada neste projeto de investigação aos sistemas IVI dos veículos da marca BMW, está dividida em quatro fases, a fase da preparação, aquisição, análise e apresentação, conforme se ilustra na figura 2.

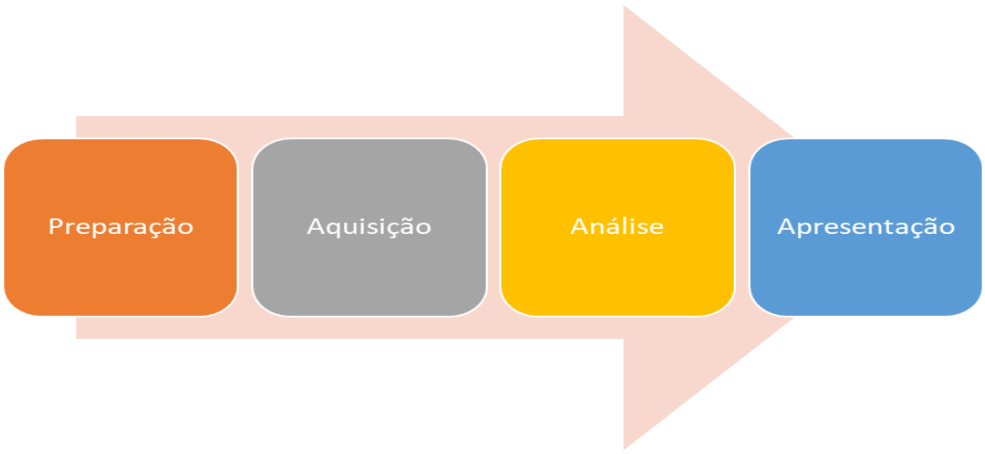


Figura 2 - Principais fases do processo de análise forense

Estas diretrizes são usadas para garantir o tratamento adequado dos casos quando se trata de provas forenses. Esta metodologia dá uma abordagem mais específica para garantir a integridade das provas e o seu manuseio adequado. É importante notar que as fases apresentadas podem ser ajustadas de acordo com o caso em si, mas é importante seguir um processo semelhante, especialmente no caso da análise forense digital, para que o processo seja legitimo.

Cada uma destas fases pode incluir um conjunto alargado de ações, em todas as fases deve-se documentar tudo que se vai fazendo na interação com o equipamento.

Esta metodologia da análise forense digital está preparada e enquadrada para análise aos sistemas IVI, como passo a descrever nas seguintes fases do processo da análise forense digital.

2.1.1. Preparação

Esta fase de preparação é a primeira fase do processo de analise forense digital. Consiste em identificar fisicamente a origem da prova digital, antes de qualquer abordagem devemos saber particularmente qual a tecnologia que vamos analisar, como funciona e eleger a melhor abordagem para a analisar.

Essa identificação pode ser feita diretamente, através do equipamento onde está armazenada, ou, caso a prova não esteja acessível, é necessária uma autorização para seu manuseio. Após a autorização, a prova é classificada e identificada.

Nesta fase de preparação existe a apreensão de um equipamento ou um dispositivo, tendo em atenção que devemos fazer sempre a custódia da prova.

A custódia da prova é um processo para preservar o cenário da prova, inclui a preservação dos dados contidos nos dispositivos e significa que em cada etapa são registados todos os passos, decisões, ferramentas usadas, as pessoas responsáveis que manusearam a prova, assim como os resultados obtidos para que sejam validados. Desta forma, é possível manter um registo no qual o dispositivo passou, por quem passou, o que foi feito, provando assim que as provas não foram alteradas, assim a custódia da prova permitiu o registo de todos os passos efetuados [12].

2.1.2. Aquisição

Durante esta fase o objetivo é a extração dos dados que estão no interior do equipamento, sendo fundamental efetuar todos os procedimentos que assegurem a preservação da prova digital.

Aquisição de dados é o processo de gerar imagens ou de outra forma de extração de informações de um dispositivo.

Uma imagem é uma cópia bit a bit de um dispositivo de armazenamento, copia todo o conteúdo e é utilizada em investigações de analise forense digital.

Deve-se usar todos os procedimentos técnico-legais para a validação da prova, nunca alterar ou danificar a fonte a ser analisada.

A proteção da informação e preservação da prova digital é alcançada através da implementação de diretrizes e procedimentos, nomeadamente instruções detalhadas sobre os direitos autorizados de recuperação de prova digital, como preparar adequadamente os sistemas de recuperação de prova, onde armazenar qualquer prova recuperada, e como documentar essas atividades para garantir a autenticidade e integridade dos dados [13].

A Figura 3, descreve o procedimento geral para extrair e analisar dispositivos com capacidade de armazenamento. O dispositivo que está a ser analisado é ligado a um bloqueador de gravação, para evitar qualquer operação de gravação que possa ser feita inadvertidamente e para impedir que a prova seja modificada. Em seguida é utilizado o software Forensic ToolKit (FTK) para extrair uma imagem bruta do armazenamento do dispositivo.

Figura 3 - Procedimento geral para extrair e analisar dispositivos eletrônicos [11]

Todas as provas devem ser validadas antes e depois da recuperação através do *hash* ou resumo digital. O resumo digital ou *hash* da prova original e de todas as cópias deve ser o mesmo para manter a integridade e a autenticidade. A integridade de uma cópia forense

criada para análise é garantida pela função *hash*. É importante notar que o mesmo algoritmo *hash* deve ser usado na imagem original da prova e também na cópia forense.

Legalmente, o processo de validação do resumo digital é denominado por "assinatura digital" conforme consta do n.º 8 do artigo 16.º, sobre a apreensão de dados informáticos, da Lei 109/2009[1], de 15 de setembro [9].

As provas em questão podem ser invalidadas pelo juiz da investigação criminal assim que apreendidas em razão do descumprimento da assinatura digital durante a fase de investigação ou mesmo durante o julgamento se tal incidente for levantado pela defesa.

No entanto, o não cumprimento não é, por si só, um motivo para invalidar as provas. Caso a assinatura digital não seja tecnicamente possível, a impossibilidade deve ser justificada em laudo próprio validado pelo tribunal, garantindo a integridade da cadeia de evidências no caso sob investigação [9].

Os dados dos sistemas IVI podem ser adquiridos, "no próprio veículo" ou "em laboratório". Os métodos usados dependem das ferramentas disponíveis para extrair os dados e do nível de análise exigido pela investigação. Níveis mais altos exigem um exame mais abrangente, habilidades adicionais e podem não ser aplicáveis ou possíveis para todos os dispositivos ou situações [6].

Os sistemas IVI dos veículos possuem diferentes tipos de hardware para armazenamento, desde o disco rígido HDD e SSD, chips de memória flash, cartões SD. Considerando as diversas tecnologias existentes, diferentes ferramentas e metodologias foram desenvolvidas de forma a extrair dados independentemente da plataforma em que estão armazenados.

Nos sistemas IVI podem ser usados diferentes tipos de aquisição, depende dos locais e o tipo de equipamento/tecnologia que se vai analisar, diferentes sistemas operativos e o método que mais se adequa à análise.

Nesta fase existem aquisições mais invasivas do que outras, por isso, a aquisição a sistemas IVI está dividida em vários métodos diferentes, aquisição lógica, física, *chip-off*, JTAG/ISP e micro read.

[1]Lei do cibercrime,Lei 109/2009
https://www.pgdlisboa.pt/leis/lei_mostra_articulado.php?artigo_id=1137A0016&nid=1137&tabela=leis&pagina=1&ficha=1&so_miolo=&nversao=#artigo

A seguir são explicados os vários métodos de análise, aquisição lógica, física, *chip-off*, JTAG/ISP e micro read, como se ilustra na figura 4.

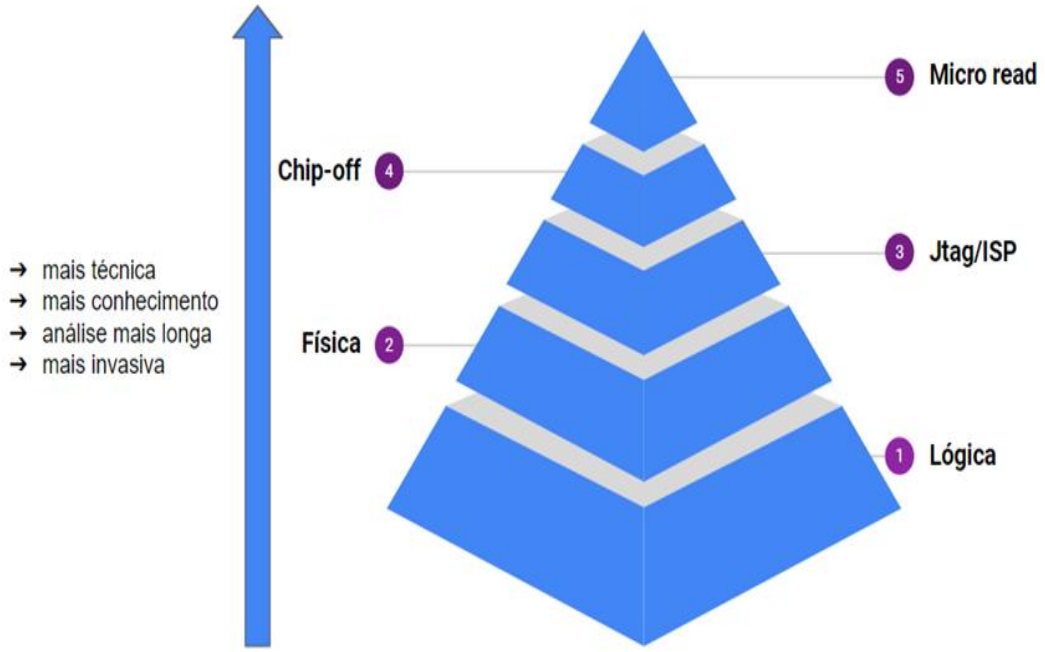


Figura 4 - Métodos de análise

- *Aquisição Lógica*

A aquisição lógica pode ser considerada uma cópia de arquivos, pastas e outras informações lógicas do sistema de armazenamento do dispositivo, como por exemplo um disco rígido. Esta aquisição permite fazer uma cópia completa de uma partição lógica de um disco rígido, cria uma cópia do conteúdo lógico dessa partição ou de um conjunto de pastas [14].

Muitas das ferramentas forenses realizam uma aquisição lógica. Uma aquisição lógica é fácil de realizar e recupera os arquivos num determinado dispositivo e não recupera ficheiros excluídos.

Esta aquisição é pouco invasiva, é o primeiro método forense, é fácil de trabalhar, todos os arquivos são listados e podem ser analisados imediatamente e é menos completa que a aquisição física.

- *Aquisição Física*

A aquisição física é realizada usando ferramentas forenses. O processo cria uma cópia bit a bit de um sistema de ficheiros inteiro. Uma aquisição física é capaz de adquirir todos os dados presentes num dispositivo, incluindo os dados excluídos na maioria dos dispositivos [15]. Permite também obter ficheiros apagados e metadados temporais adicionais.

Na criação de imagens físicas de um dispositivo de armazenamento, existem vários formatos disponíveis para o ficheiro de destino da cópia, de que se destacam os mais utilizados para o efeito: E01 e Raw (dd). Os formatos E01 e Raw, são os formatos mais usados na análise forense digital. O formato Raw permite a criação de uma cópia bit a bit exata de um dispositivo de armazenamento. A imagem resultante é uma réplica exata do dispositivo de armazenamento original. O formato E01 permite a criação de uma imagem do dispositivo de armazenamento de forma compactada. Ambos os formatos têm as suas vantagens e desvantagens, o formato E01 suporta recursos avançados de compressão e não é compatível com todas as ferramentas de análise forense, enquanto o formato Raw (dd) não possui recursos avançados de compressão e é compatível com a maioria das ferramentas de análise forense. Em resumo, a escolha entre o formato E01 e o formato Raw (dd) depende do software de análise forense que vamos usar, depende dos recursos que necessitamos e das exigências do caso específico.

O conteúdo recuperado nesta aquisição é superior ao de uma aquisição lógica. Portanto mais dados úteis podem ser recuperados.

- *Chip-off*

Chip-off é uma técnica avançada de extração de dados que requer a remoção física de chips de memória do dispositivo, como por exemplo, Processor Chips, NAND Flash e EEPROM. Estes são alguns exemplos de chips que podem ser analisados, existe muitos outros tipos de chips que podem ser removidos e analisados. Esta técnica deve ser usada apenas em último recurso. É frequentemente usada nos casos em que o dispositivo está danificado ou criptografado e os outros métodos de análise não conseguem extrair os dados.

A técnica *chip-off* não se limita apenas a telefones, mas também pode ser usado noutros dispositivos, como no caso dos sistemas IVI. O processo da técnica *chip-off* consiste nas seguintes etapas:

1. Remover o chip do dispositivo e limpar o chip,
2. Aquisição de imagem com o uso de equipamentos especializados,
3. Análise da imagem com ferramentas forenses padrão.

Esta técnica é muito usada para analisar sistemas IVI. O investigador deve ser experiente e habilidoso, especialista na área, porque o chip pode ser destruído durante o processo, uma vez que é uma peça muito sensível e fácil de estragar [16]. As informações obtidas da memória estão num formato bruto, devem ser analisadas e interpretadas. O método chip-off é o preferido em situações onde é importante preservar o estado de memória exatamente como ele existe no dispositivo [15].

- *Join Test Action Group (JTAG)/In System Programming (ISP)*

O método JTAG é um processo que envolve a ligação às portas de acesso de teste (TAP) através de solda, a seguir é ligado a uma JTAG Box, permitindo adquirir os dados brutos armazenados no chip de memória, que por sua vez, permite obter uma imagem física completa do dispositivo [15]. Existem no mercado vários modelos de JTAG Box com capacidade de executar este método, sendo as mais conhecidas a Easy JTAG, Z3X, Riff e ATF. A Figura 5 exemplifica o método JTAG e a figura 6 menciona as portas de acesso de teste de um dispositivo.

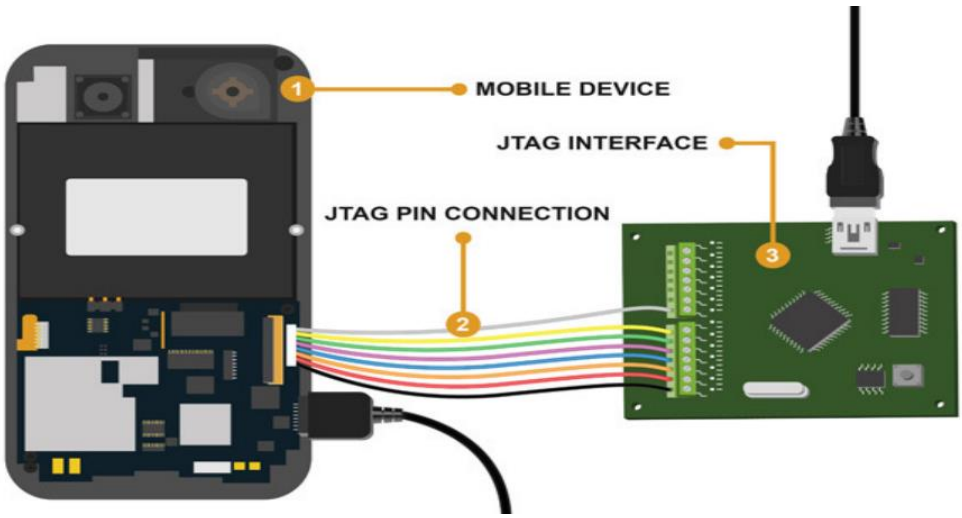


Figura 5 - Execução do método JTAG [17]

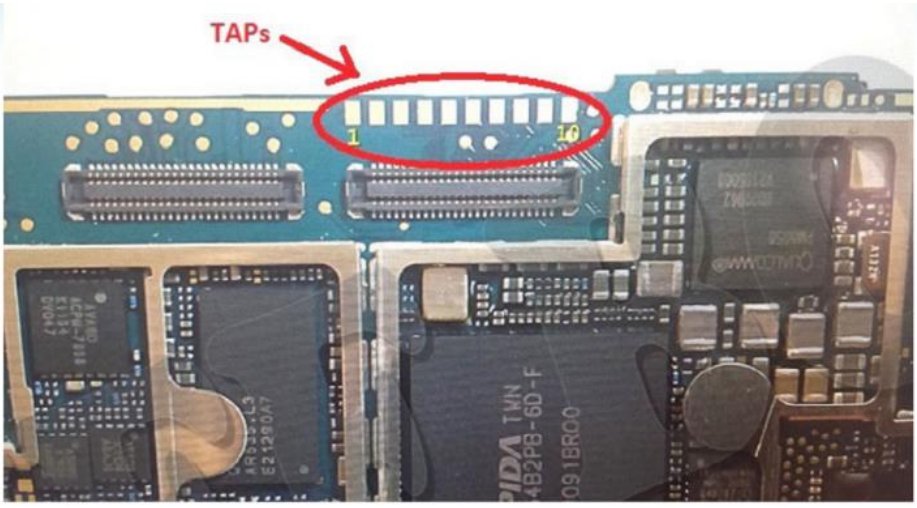


Figura 6 - Portas de acesso de teste (TAP)

O método ISP é uma técnica ao semelhante ao JATG, porém mais rápida, cuja principal diferença é que no método ISP, a ligação é feita diretamente à TAP da memória flash, e pode ser usado quando o dispositivo não é compatível com o método JATG. Neste método é necessário conhecer os TAPs que se ligam à memória flash. O objetivo deste método é recuperar o conteúdo completo da memória do dispositivo. É necessário o uso de uma Box para fazer a ligação. A execução do método ISP é um processo que requer vários passos, deve-se ligar aos TAPs de acesso à memória flash, depois são soldados nesses TAPs, por fim ligamos os TAPs à Box [18]. A Figura 7 exemplifica as soldas na Box e nos TAPs das memórias flash.

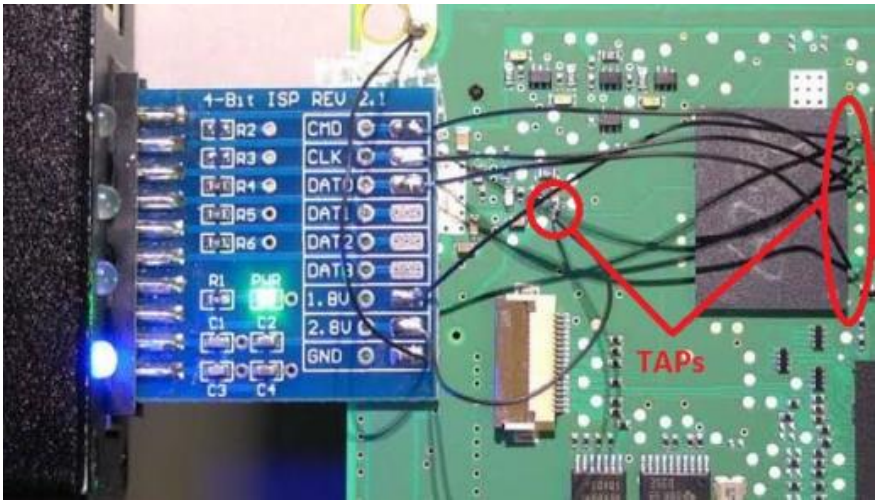


Figura 7 - Placa pronta para execução do método ISP [19]

- *Micro-read*

O processo envolve a visualização e interpretação manual dos dados vistos no chip de memória. O examinador usa um microscópio eletrônico e analisa as portas físicas do chip e em seguida, traduz o status da porta para 0 e 1 para determinar os caracteres ASCII resultantes. Todo o processo é demorado e caro, e requer muito conhecimento em memória flash e sistemas de ficheiros. Devido aos aspetos técnicos extremos envolvidos em micro read, será apenas usado em casos em que não seja possível usar as outras técnicas de extração [15].

2.1.3. Análise

Esta fase está relacionada com análise e investigação dos dados existentes na fase de aquisição. Uma vez que a informação já foi extraída, deve ser identificada a prova. Nesta investigação deve ser usada uma abordagem metódica para chegar a conclusões apropriadas com base nos dados disponíveis ou determinar que nenhuma conclusão ainda pode ser retirada [20].

A análise de dados pode ser realizada com recurso a diversas ferramentas. Esta fase está mencionada com mais detalhe no capítulo 4, onde serão abordados os vários sistemas IVI da BMW.

2.1.4. Apresentação

A apresentação é a última etapa no processo de análise forense digital. É nesta fase que se produzem os relatórios técnicos, onde serão mencionados os nomes dos equipamentos analisados, as ferramentas que foram usadas na análise, os métodos de recolha utilizados, quais as medidas tomadas na proteção e preservação das provas, bem como, os resultados obtidos da investigação e conclusões obtidas na análise das provas, por isso, o grande objetivo desta fase é a demostração probatória dos dados que estão em investigação [9].

É fundamental que estes relatórios sejam redigidos de forma clara, concisa e precisa, para que todas as partes envolventes na leitura do mesmo percebam o seu conteúdo.

2.2.Ramos da Análise Forense Digital em Veículos

A análise forense em veículos é um ramo da análise forense digital. Há várias partes do veículo que podem ser analisadas, contendo dados que podem ser usados durante uma

investigação. As informações fornecidas pelas diferentes partes do veículo durante uma análise, podem ajudar a responder a várias questões importantes para a investigação a nível criminal e acidentes de viação.

A investigação nos sistemas IVI é a parte do veículo que é investigada e analisada neste projeto. Na atualidade todas as marcas de veículos incorporam um sistema IVI, o que terá um grande impacto nas investigações no futuro.

A análise forense digital nos veículos, na figura 8 ilustra os tipos de análise forense que podem ser realizados num veículo.

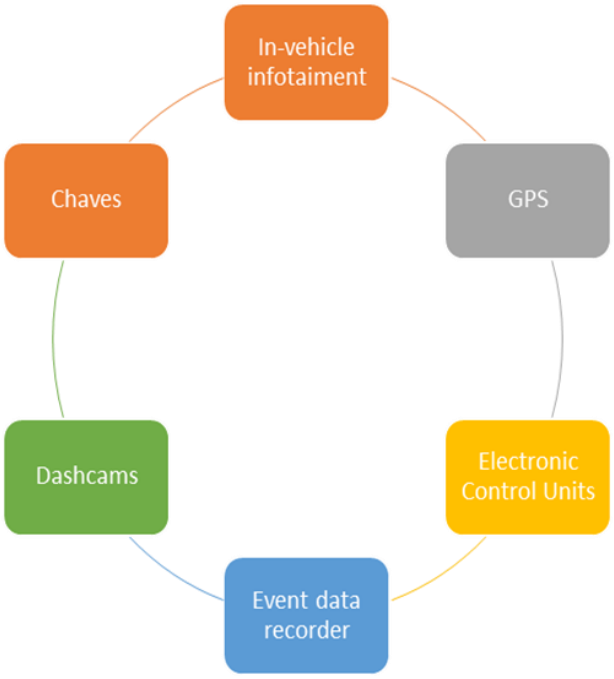


Figura 8 - Ramos da análise forense em veículos

2.2.1. Análise Forense Digital a Sistemas IVI

A análise forense digital nos sistemas IVI é importante e facilita a investigação para os investigadores, do ponto de vista forense estes dados podem ajudar a resolver crimes e acidentes. Este projeto é sobre análise forense a sistemas IVI, que será mais desenvolvido noutros capítulos deste projeto.

Estes sistemas têm como objetivo tornar as experiências de condução amigáveis e proporcionar ao condutor e respetivos passageiros uma ligação constante ao veículo [21].

A palavra "Infoentretenimento" é um termo inglês, que nasce da união de duas palavras, “informação” e “entretenimento” [22].

Os sistemas IVI são uma unidade central de comunicação entre o veículo e os seus ocupantes. A evolução da tecnologia permitiu a implementação de sistemas de infoentretenimento nos veículos, permitindo aos condutores uma maior interatividade, isso teve um grande impacto no desenvolvimento dos veículos e tecnologias incorporadas que os tornaram mais utilitários. Em suma os veículos são exatamente computadores.

Nos sistemas IVI, os dados podem ser transmitidos através de Bluetooth, Wi-Fi, USB, ou dados móveis. Os sistemas IVI comunicam com vários dispositivos como telefones e tablets.

Os sistemas IVI podem ser fabricados pelas próprias marcas de veículos, embora a maioria seja comprado a um fornecedor de tecnologia, isso significa que não foi estabelecido nenhum padrão para o desenvolvimento dos sistemas IVI.

A quantidade de dados armazenados nesses sistemas IVI pode variar significativamente de um fabricante para outro. O mesmo fabricante pode fornecer muitos sistemas IVI diferentes em toda a gama dos modelos de veículos [5].

Um sistema IVI consiste em muitos componentes de hardware e software ligados entre si, a sua arquitetura geral pode ser descrita com um conjunto de camadas, desde a camada de hardware, na parte inferior, até a Human–Machine Interface (HMI), na parte superior.

Os sistemas IVI oferecem recursos de entretenimento e informações, possuindo diferentes tipos de dados, tais como:

- Dados referente ao Vehicle Identification Number (VIN);
- Números de serie;
- ID dispositivo;
- Dados de navegação GPS;
- Tracklogs;
- Trackpoints;
- Waypoints;
- Localizações guardadas;
- Localizações recentes;
- Vídeo;

- Música;
- Imagens;
- Contatos;
- Mensagens de texto SMS;
- Chamadas;
- Dados MAC address Bluetooth;
- EMEI;
- IMSI;
- Integração com smartphones através do Android Auto e CarPlay da Apple;
- Instalação de aplicações;
- Dados (Dash cam);
- Dados de temperatura;
- Dados dos sensores de proximidade;
- Logs de Velocidade;
- E-mail;
- Tarefas;
- Calendário.

Como ilustra na figura 9, hoje os veículos modernos possuem diversos serviços, podemos lhes chamar computadores sobre rodas.

Figura 9 - Serviços presentes nos sistemas IVI

Os sistemas IVI permitem aos condutores receber orientações e localizações pela navegação GPS, assistir a filmes ou ouvir música, efetuar chamadas, receber SMS e E-Mail.

Encontramos vários sistemas IVI instalados nos veículos, incluindo BMW, Ford, Chrysler, Honda, Mazda, Subaru, Toyota, Tesla, Audi, Porsche, SEAT, Citroen, Renault, Peugeot, Mercedes, Volkswagen e outros. Cada fabricante de automóveis tem o seu próprio sistema IVI e existem diversos sistemas operativos, tais como:

- Blackberry QNX;
- Automotive Grade Linux;
- Android Automotive;
- Windows.

Neste momento existem marcas de veículos que usam mais que um sistema operativo, em diversos modelos dentro da mesma marca. O sistema IVI é constituído por um ecrã na consola central, essa interface é exposta na Figura 10.

Figura 10 - Display sistema IVI BMW com Sistema Operativo QNX [23]

A Google uniu-se a vários fabricantes de automóveis para estabelecer o Open Automotive Alliance, que tem o objetivo de desenvolver uma plataforma comum entre sistemas de infoentretenimento e android. Esta aliança permitiu que se desenvolvesse o Android Auto. A Apple também fez uma aliança e desta forma foi concebido o Apple CarPlay [21].

A interligação de dispositivos móveis tornou-se nos últimos anos uma componente importante nos sistemas IVI, visto que a Apple e o Android que também possuem sistemas de infoentretenimento. O Android Auto e o Apple CarPlay são dois sistemas operativos integrados com o sistema IVI do veículo, que permitem controlar algumas das funções do smartphone no display do sistema IVI do veículo. Para isso, basta ligar o smartphone ao veículo através do sistema Bluetooth ou do cabo USB para começar a utilizar de imediato as diversas funcionalidades do smartphone no sistema IVI do veículo, como se ilustra nas figuras 11 e 12.

Figura 11 - Android auto

Figura 12 - Apple CarPlay

Os sistemas IVI atualmente contêm inúmeras informações sobre os utilizadores, como o Bluetooth dos telefones sincronizados ao veículo, histórico de chamadas, registos de mensagens, e-mails, contatos, dados do modelo e marca do telefone, IMSI, IMEI, dados de

navegação GPS, como localizações e percursos efetuados. O sistema IVI também contém informações sobre os arquivos de música e imagens. Essas informações podem ser cruzadas e examinadas durante uma investigação forense para fornecer detalhes históricos sobre o uso do veículo e dos seus ocupantes.

Os dados do Bluetooth, permitem saber o *mac address*, os dispositivos que estiveram ligados ou sincronizados com o veículo, o modelo e o número de série do telefone.

Os contatos exibem informações como, nome, número de telefone, e-mail, residência, permitindo correlacionar o utilizador do veículo a um telefone.

O registo de chamadas, permite visualizar o histórico de chamadas, o registo de mensagens e possibilita o histórico da troca de mensagens do dispositivo ligado ao sistema IVI.

Os dados tipicamente são armazenados em bases de dados SQLite, que são muito importantes a nível forense. Os dados dos sistemas IVI são importantes para reconstruir acidentes ou investigar um determinado crime envolvendo um automóvel, investigar roubos e furtos em veículos, combater o terrorismo e o crime organizado, rastrear o percurso de um determinado suspeito, verificar o modus de operandi do suspeito e identificar o condutor. Os dados podem ser benéficos para os profissionais da investigação criminal e para os peritos de acidentes, estes dados contribuem para obter as provas seguintes [24]:

- Vincular um veículo específico a um determinado local numa data e hora específica;
- Identificar o percurso de uma viagem de um veículo para uma determinada data e hora;
- Determinar o veículo num determinado local específico;
- Determinar quanto tempo alguém ficou num determinado local;
- Estabelecer um cronograma geral de eventos para um determinado período sob investigação;
- Vincular um determinado indivíduo a um veículo numa determinada data e hora;
- Identificar fontes potenciais de desatenção do condutor;
- Identificar os contatos e redes potenciais para indivíduos sob investigação;

2.2.2. Análise Forense Digital a Sistemas de Posicionamento Global (GPS)

O GPS permite que os dispositivos sejam localizados, a navegação por mapas, assim como agregar dados de localização. Do ponto de vista forense os dados de GPS podem trazer

inúmeras provas que podem vir a ser úteis. Os dados de GPS podem ser divididos em duas categorias, os dados do sistema e os dados do utilizador. Os dados do sistema guardam pontos de localização assim como um possível registo dos locais onde passou um determinado utilizador (TrackPoints). Estes pontos de utilização são automaticamente criados pelo sistema, como dados do utilizador podemos considerar os (WayPoints), dados criados pelo utilizador ou os pontos de interesse (Points). Neste caso, estes pontos de localização podem indicar localizações que o utilizador pesquisou ou frequentou. [2].

2.2.3. Análise Forense Digital Electronic Control Unit (ECU)

Ao remover a ECU ou aceder a esses dados através da porta On-Board Diagnostic (OBD), os investigadores forenses podem recuperar informações reais dos quilómetros do veículo, Vehicle Identification Number (VIN) e números de série. Essas informações podem ser usadas para identificar formalmente o veículo. Além disso, permitem visualizar um histórico de erros, o horário em que foram acionados e os quilómetros. Alguns veículos também permitem que os dados do acidente sejam recuperados juntamente com um histórico de erros e eventos, como as informações dos travões. A ECU também contém memória externa, que pode ter interface para obter informações [25]. Na figura 13, ilustra um ECU de um veículo.

Figura 13 - Exemplo ECU [26]

2.2.4. Análise Forense Digital Chaves dos Veículos

As chaves dos veículos modernos contêm inúmeras informações sobre o veículo, como a marca, número VIN, o número de chaves associadas ao veículo, ID da chave, a última leitura dos quilómetros e o estado do combustível [25].

As chaves são cada vez mais uma fonte potencial de evidências digitais úteis. Controlo remoto de sistemas de entrada sem chave e sistemas de chave inteligente pode potencialmente armazenar uma variedade de dados [5]. A figura 14 ilustra os dados recolhidos de uma análise a uma chave de um veículo da BMW.

Figura 14 - Dados de uma chave de um veículo BMW

2.2.5. Análise Forense Digital Event Data Recorder (EDR)

O EDR é conhecido como uma "caixa preta", o EDR é acionado quando o veículo está envolvido numa colisão. Geralmente, regista 5 segundos de dados antes do incidente, que podemos chamar o pré-acidente, bem como dados durante o curso da colisão. O EDR pode fornecer uma excelente fonte de evidências para investigadores de acidentes de viação [5].

Permite obter informações muito importantes como a velocidade do veículo, velocidade do motor, uso do cinto de segurança, se o airbag foi acionado, posição e ângulo do acelerador, se acionou os travões e a hora do evento.

2.2.6. Análise Forense Digital Dashcams

As Dashcams são vitais para ajudar a investigar incidentes dos veículos. Muitos veículos agora têm Dashcams, uma câmara introduzida no veículo que grava imagens de vídeo e áudio. Tem um valor significativo de evidência em forense digital, pois fornecem dados de GPS, dados temporais, dados de velocidade do veículo, áudio, vídeo e imagens fotográficas [27].

3. Tecnologias

Este capítulo fornece uma visão geral das tecnologias e ferramentas que foram usadas neste projeto. Apresenta adicionalmente a única ferramenta paga no mercado, que permite analisar os sistemas IVI.

3.1. Ferramenta Berla iVe

O software Berla iVe[2] é uma ferramenta de análise forense digital a sistemas IVI, é a única solução existente no mercado. É uma ferramenta paga, permite recolher dados dos utilizadores dos veículos e permite aos investigadores forenses analisar de forma rápida e intuitiva.

Permite analisar uma grande quantidade de dados, como destinos recentes, locais favoritos, registos de chamadas, listas de contatos, mensagens, SMS, e-mails, fotos, vídeos e o histórico de navegação de GPS.

Atualmente, o hardware iVe e o software iVe suportam a extração de dados por meio de aquisições lógicas e físicas de vários veículos.

Esta ferramenta permite:

- Aquisição Lógica e Física;
- Recuperação de dados excluídos;
- Visualizar e analisar resultados;
- Mapa de navegação GPS;
- Pesquisar e filtrar dados;
- Gerar relatórios;
- Exportar dados nos formatos GPX, KML e CSV.

A ferramenta Berla iVe permite identificar quais os veículos que são suportados por esta ferramenta, permite identificar os sistemas IVI instalados nos veículos e determina o tipo de dados que podem ser obtidos. Durante uma aquisição pode exigir que os sistemas IVI sejam removidos do veículo, desmontados ou executados no próprio veículo. Em ambos os casos,

[2] https://berla.co/ecosystem/

o hardware de aquisição pode ser ligado diretamente aos sistemas IVI no próprio veículo, através da ligação de um cabo USB ou remoção do sistema IVI veículo.

A ferramenta Berla iVe permite adquirir uma imagem e descodificar os dados. Pode recuperar informações eliminadas, analisar os dados e apresentar os dados aos investigadores de forma intuitiva. No entanto, todo o processo é especificamente projetado para ser não destrutivo, e o sistema IVI pode ser colocado de volta no veículo e permanecer totalmente operacional [28].

A ferramenta Berla iVe, tem a capacidade de encontrar dados como:

- Tempo;
- Tráfego;
- Aplicações instaladas;
- Dispositivos ligados ao veículo;
- Telefones;
- Dados de Navegação GPS;
- Tracklogs e Trackpoints;
- Locais guardados;
- Destinos anteriores;
- Informação do Dispositivo;
- IDs do dispositivo;
- Chamadas;
- Contactos;
- SMS;
- Áudio;
- Vídeo;
- Imagens;
- Eventos;
- Abertura/Fecho de Portas;
- Luzes acesas/desligadas;
- Ligações Bluetooth;
- Ligações Wi-Fi;
- Ligações USB;
- Leituras do Odómetro;

- Gravação áudio;
- Velocidade do veículo.

A Berla já permite a análise de sistemas IVI de veículos de várias marcas, incluindo BMW, Cadillac, Chevrolet, Chrysler, Dodge, FIAT, Ford, GMC, HUMMER, Jeep, Maserati, MINI, Mercedes, Opel, Pontiac, Rolls-Royce, SEAT, Skoda, Toyota, Vauxhall e Volkswagen.

Berla iVe Ecosystem é uma coleção de ferramentas que apoia os investigadores em todo o processo forense de veículos com uma aplicação móvel para identificação de veículos, um kit de hardware para aquisição e um software forense para análise de dados, que é composto por:

- IVe Mobile;
- iVe Toolkit;
- iVe Desktop.

3.1.1. iVe mobile

O iVe Mobile é uma aplicação mobile, que pode ser instalada no android e iOS. Permite que os investigadores identifiquem os veículos suportados pelo iVe, permite saber quais os sistemas IVI que estão instalados e que tipo de dados podem ser recuperados. Fornece instruções para localizar e remover os sistemas IVI dos veículos.

O iVe mobile fornece ainda:

- Ferramentas de pesquisa do veículo;
- Guias de identificação do sistema IVI;
- Instruções de remoção do sistema IVI;
- Descrições do Método de Aquisição;
- Visualizador de conteúdo de dados adquiridos;
- Recebe notificações quando são adicionados novos veículos e instruções de aquisição.

A Figura 15, demostra a aplicação iVe mobile que permite escolher a marca do veículo, ano e modelo.

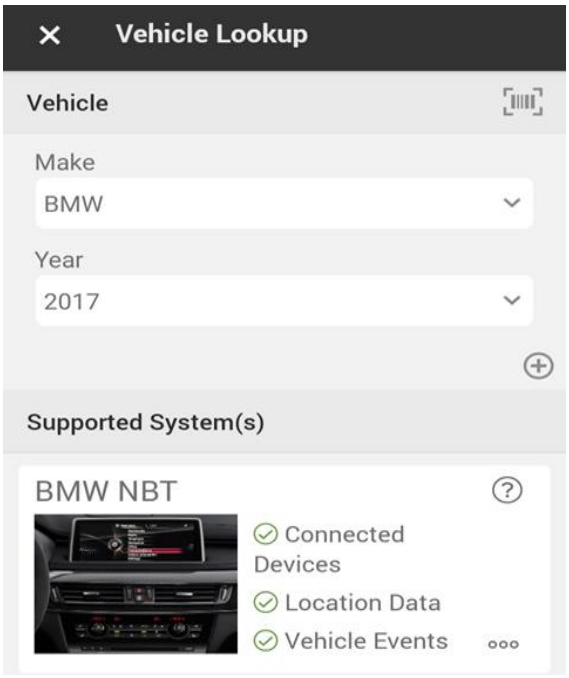


Figura 15 - iVe mobile escolher marca, modelo, ano [4]

A figura 16, ilustra que dados possui o modelo do sistema IVI que foi escolhido anteriormente e fornece um guia de remoção do sistema IVI.

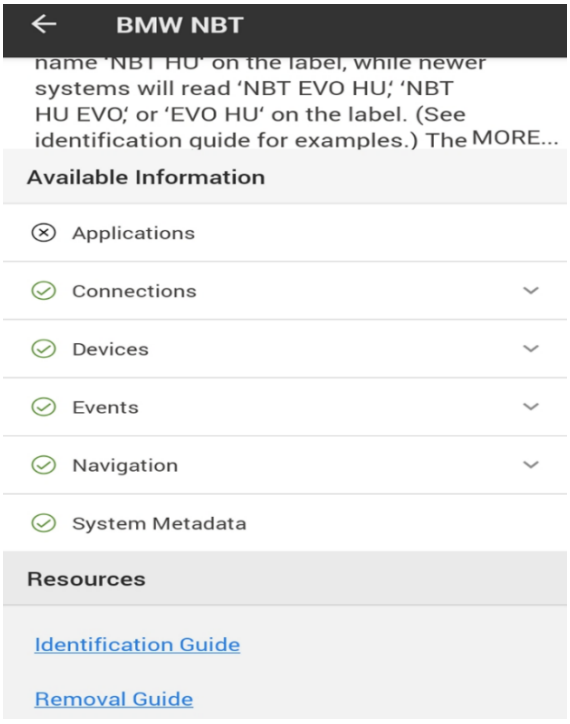


Figura 16 - iVe mobile tipo de dados e remoção [4]

3.1.2. Hardware KIT

O iVe Toolkit é um conjunto de cabos desenvolvidos especificamente para vários sistemas IVI. O kit de ferramentas inclui ferramentas para ajudar a remover os sistemas IVI de um veículo. Os cabos são usados em conjunto com o software iVe para adquirir os dados.

3.1.3. Software forense

Berla iVe Desktop é uma aplicação que só funciona no sistema operativo Windows, é usado para analisar dados e recuperar informações excluídas. O iVe Desktop inclui um conjunto de ferramentas de análise, relatórios, exportação de dados, pesquisa e análise de linha do tempo. A figura 17 e 18, demonstra a layout do iVe Desktop.

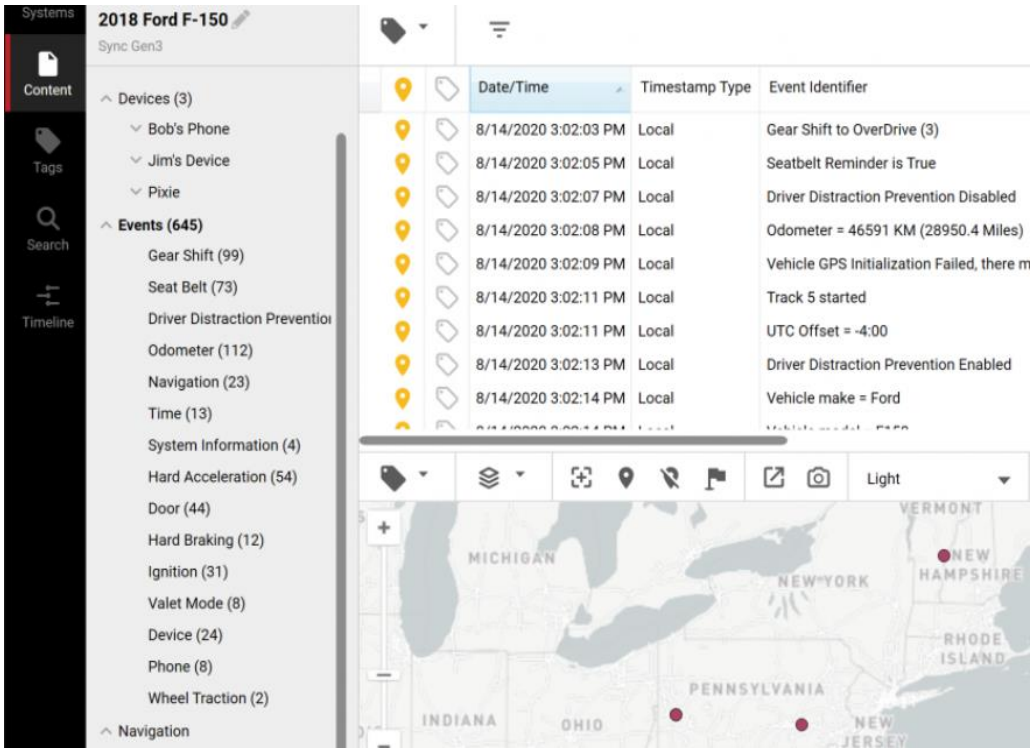


Figura 17 - Análise iVe Desktop [28]

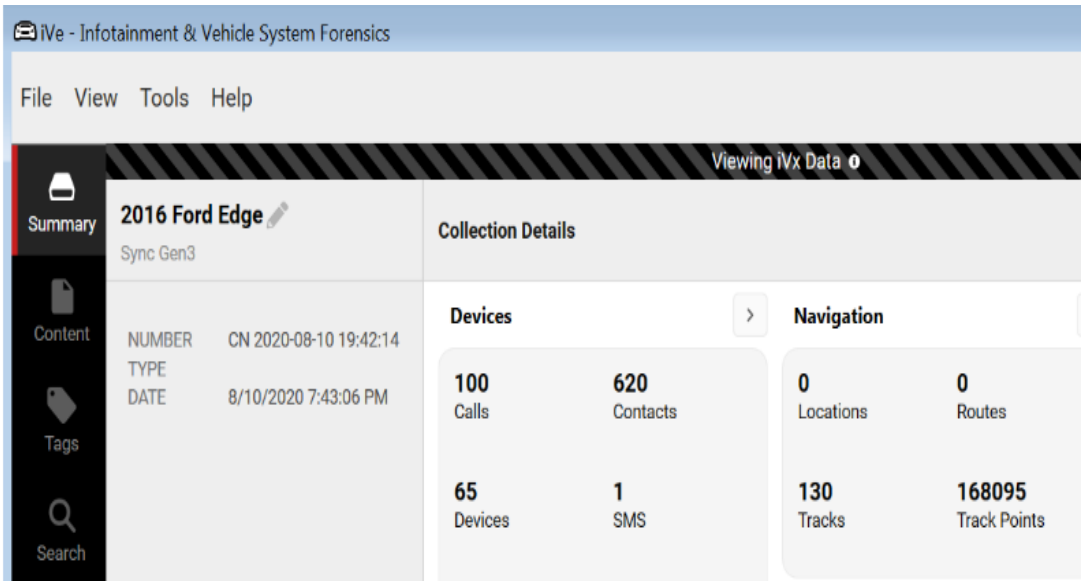


Figura 18 - Análise iVe Desktop [28]

3.2. QNX Neutrino Realtime Operation System (RTOS)

Os sistemas IVI da marca BMW analisados neste projeto possuem sistema operativo QNX. O sistema operativo QNX foi desenvolvido pela Blackberry, e é um dos mais usados no mercado automóvel. No que diz respeito aos veículos, a sua primeira versão foi lançada em 1982.

A Blackberry criou a QNX CAR para sistemas IVI, que é um conjunto de componentes de software com todas as funções necessárias para apoiar a indústria automóvel a criar sistemas de infoentretenimento.

Devido às suas características o sistema operativo QNX é robusto e tem uma grande proteção contra erros de programação [29].

Desde 1980 muitas empresas usaram o sistema operativo QNX RTOS, para garantir a combinação ideal de desempenho, segurança e confiabilidade em sistemas de missão crítica.

O sistema operativo QNX é muito usado devido às suas características, como se ilustra na figura 19.

Figura 19 - Características SO QNX

O sistema operativo QNX é usado em milhões de veículos e diferentes marcas de automóveis, desde a BMW, Ford, VolKswagen, Saab, Skoda, Aptiv e Bosch [30].

Atualmente o sistema operativo QNX NEUTRINO RTOS, está na versão 7.1. Esta versão foi lançada em 23-06-2020. O Sistema operativo QNX, está presente em muitos serviços existentes nos veículos como se ilustra na figura 20 [31].

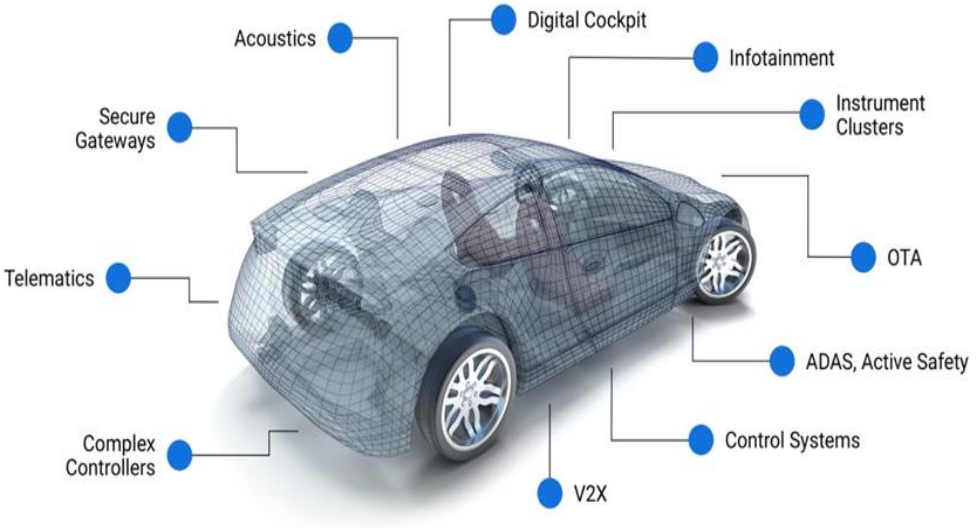


Figura 20 - Serviços SO QNX [32]

3.2.1. Sistema de ficheiros QNX

O sistema de ficheiros QNX, é um sistema de ficheiros proprietário, muito utilizado numa ampla gama de dispositivos, incluindo nos sistemas IVI instalados nos veículos. O sistema de ficheiros define a maneira como os arquivos são nomeados, armazenados, organizados e acedidos em volumes lógicos. Usa os conceitos de diretorias e arquivos para organizar e armazenar dados [33].

QNX é o sistema de ficheiros existentes nos sistemas IVI da BMW, em resumo, o sistema de ficheiros QNX, é confiável, possui alto desempenho, suporta grandes partições, faz a gestão de forma eficiente de espaço em disco, é tolerante a falhas, com desempenho rápido de leitura/gravação e oferece recursos avançados de gestão de arquivos e disco.

3.2.2. Partições do sistema operativo QNX

Um disco rígido está dividido em várias partes, chamadas de partições. As partições são usadas para separar o disco em unidades de armazenamento lógicas [14]. No sistema operativo QNX, as partições são usadas para separar diferentes tipos de dados, garantir a segurança e estabilidade do sistema. Cada partição é isolada do resto, o que significa que um problema numa determinada partição não afetará outras partições ou o sistema como um todo.

No disco rígido dos sistemas IVI BMW, o disco rígido é constituído por várias partições, cada uma destinada a um tipo específico de dados, como sistema operativo, dados do utilizador e configurações do sistema.

Além disso, o sistema operativo QNX usa o modelo de partição hierárquico, que permite que as partições sejam subdivididas em várias partições. Isto é particularmente útil em sistemas complexos, como os encontrados nos veículos, onde vários componentes precisam de ser executados em paralelo sem interferir uns com os outros. No sistema operativo QNX cada partição deve receber um tipo de sistemas de ficheiros reconhecido pelo sistema operativo, neste caso o sistema operativo QNX usa o sistema de ficheiros qnx6 ou qnx4 para fazer a gestão de um disco rígido.

3.3.Bases de dados SQLite

As bases de dados existentes nos sistemas IVI da BMW analisados neste projeto são do formato SQLite, que permite armazenar informações sobre os contatos, registo de chamadas,

mensagens, dados do telefone e músicas. SQLite, é uma biblioteca que pode ser embutida em aplicações, que implementa um mecanismo de base de dados SQL transacional independente, sem servidor e sem configuração. SQLite está presente em muitas aplicações, smartphones e sistemas operativos.

SQLite lê e grava diretamente em ficheiros de disco comuns. Uma base de dados SQL é constituída por várias tabelas e está contida num único arquivo de disco. O formato do arquivo da base de dados é multiplataforma, pode copiar livremente uma base de dados entre sistemas de 32 bits e 64 bits ou entre arquiteturas big-endian e little-endian. Estas características tornam o SQLite uma escolha popular como um formato de arquivo de aplicações [34].

3.4.Ferramentas usadas neste projeto

As ferramentas de análise forense digital são uma grande ajuda para os investigadores de análise forense digital. Este projeto permitiu identificar várias ferramentas e processos, que podem ser aplicadas aos sistemas IVI.

3.4.1. Kali Linux

Kali Linux é uma distribuição Linux baseada na distribuição Debian, direcionada para tarefas de segurança da informação, pentest, análise forense digital e engenharia reversa. Kali Linux está disponível gratuitamente. Foi usado para montar as partições do sistema operativo QNX. Com os recursos do Kali Linux foi possível aceder às pastas e ficheiros contidos nas várias partições, o que será explicado na secção 4.5 do capítulo 4.

Instalação é possível através de imagens prontas a funcionar no VMWARE ou no VirtualBox, a instalação é bastante fácil, basta importar a imagem para as máquinas virtuais. Neste caso o utilizador tem vantagem de mudar o hardware virtual, pode, por exemplo diminuir ou aumentar a RAM, mudar a forma de funcionamento da placa de rede e aumentar o espaço do disco. A análise foi feita num computador com o sistema operativo Windows 10 a correr uma VM Virtual Box com uma imagem do Kali Linux.

3.4.2. VM Virtual Box

As máquinas virtuais, não são diferentes de qualquer outro computador. Têm CPU, memória, discos para armazenar os seus ficheiros e podem ligar-se à Internet. As máquinas virtuais

são, por norma, encaradas como computadores virtuais ou computadores definidos pelo software dentro de servidores físicos, existindo apenas como código [35].

Máquina virtual VirtualBox é um software de virtualização gratuito e de código aberto que permite criar e executar máquinas virtuais num computador. Foi desenvolvido pela Oracle e está disponível para Windows, macOS, Linux e Solaris.

VM Virtual Box, estende os recursos do computador existente para que ele possa executar vários sistemas operativos, dentro de várias máquinas virtuais, ao mesmo tempo. Por exemplo, pode executar o sistema operativo Linux no Windows, executar Windows Server em servidor Linux. Permite instalar e executar quantas máquinas virtuais desejar. Os únicos limites práticos são o espaço em disco e a memória [36].

3.4.3. Db browser SQLite

DB Browser for SQLite é uma ferramenta open source, com interface gráfica para criar, editar base de dados compatíveis com SQLite. DB Browser for SQLite é para utilizadores e programadores que desejam criar, pesquisar e editar base de dados. Permite criar e compactar arquivos de base de dados, criar, definir, modificar e excluir tabelas e permite ainda realizar consultas SQL [37]. O DB Browser for SQLite neste projeto foi usado para consultar e visualizar as diversas bases de dados dos sistemas IVI analisados e permitiu saber a estrutura e quais os dados incluídos nas bases de dados, conforme ilustram as figuras 21 e 22.

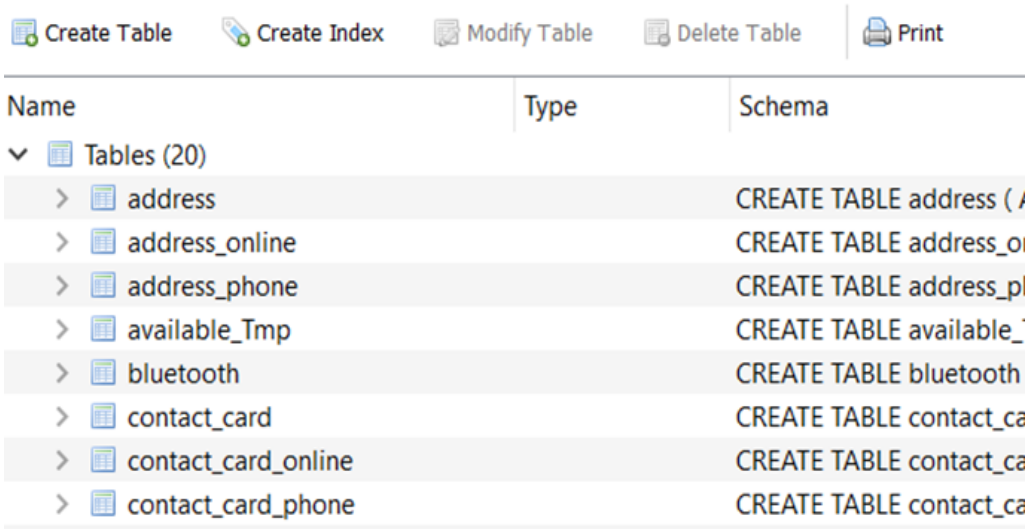


Figura 21 - Base de dados contactbook_20120928

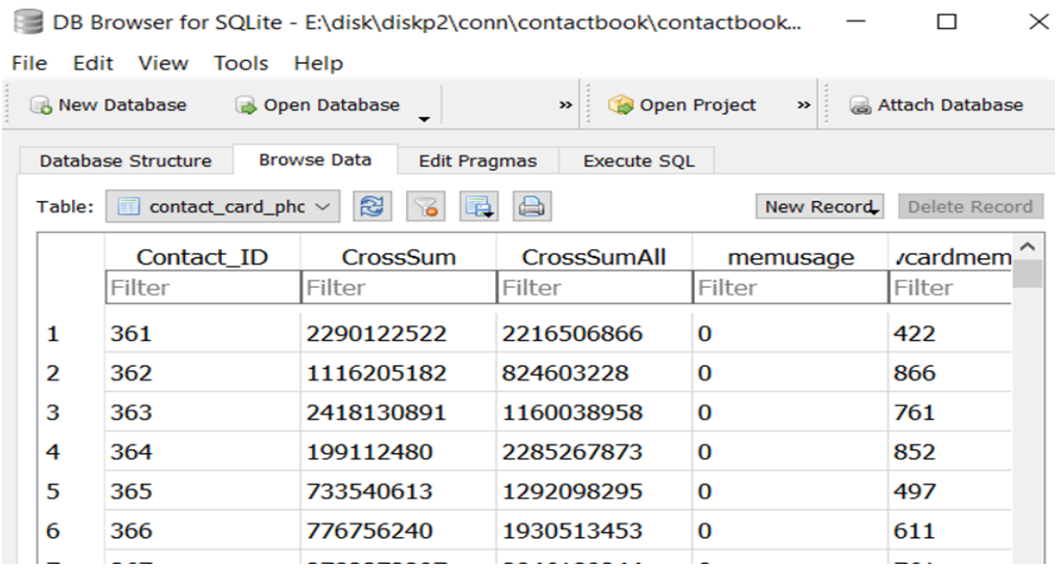


Figura 22 - Dados da tabela contact_card_phone

O uso desta ferramenta foi muito importante para ajudar a elaborar os dois plugins ingest module para a ferramenta Autopsy.

3.4.4. Qphotorec

QPhotoRec é um programa que permite a recuperação de ficheiros, incluindo ficheiros de áudio e vídeos, documentos, fotos, base de dados, entre outros ficheiros, contidos em dispositivos de armazenamento. Este programa é muito importante para encontrar ficheiros excluídos [20]. Este programa foi usado para recuperar ficheiros excluídos dos discos rígidos HDD dos sistemas IVI que foram analisados neste projeto.

3.4.5. AccessData Ftk Imager

O FTK Imager é um software open source da AccessData, permite criar imagens sem alterar os dados originais. Neste projeto foi usada a versão 4.2.0.13. A ferramenta FTK Imager também fornece a função de verificação de integridade, que gera um valor *hash*.

A imagem forense é uma das etapas mais importantes na análise forense digital, é o processo de fazer uma cópia de todo o disco rígido [38]. Esta ferramenta foi usada para criar as imagens físicas dos discos rígidos HDD dos sistemas IVI analisados neste projeto.

3.4.6. Autopsy

Neste projeto foi usada a ferramenta Autopsy para analisar os dados das imagens efetuadas aos sistemas IVI da marca BMW, para isso foram desenvolvidos 2 plugins ingest module, o que é explicado na secção 6.1 do capítulo 6.

O Autopsy é uma plataforma de análise forense digital open-source e é utilizado por investigadores no contexto de investigações no âmbito da análise forense digital. O Autopsy é uma ferramenta fácil de usar, intuitiva, útil na análise de imagens de discos, bem como qualquer outro tipo de dispositivos que contém armazenamento [39].

Esta ferramenta contém uma interface gráfica para as ferramentas de análise através da linha de comando do The Sleuth Kit [40].

Este software pode ser utilizado em sistemas operativos Windows, Linux e Mac OS. A instalação é simples, rápida e disponível no website oficial[3], tal como foi realizado neste projeto.

O Autopsy permite a integração de plugins ingest module e report, sendo permitido o desenvolvimento de módulos em Java ou Python. Os plugins ingest module desenvolvidos permitem que sejam personalizados para diversas situações. É de ter em conta que existem vários tipos de módulos, os quais podem ser aplicados a locais diferentes na ferramenta de acordo com o objetivo implícito.

É de mencionar que neste projeto foi utilizada a versão 4.18.0 num sistema operativo Windows 10. A ferramenta Autopsy neste momento encontra-se na versão 4.20.

Para uma melhor compreensão desta ferramenta [11], é importante compreender seus principais conceitos:

- O Autopsy possui quatro tipos de módulos, insgest modules, report modules, contente viewers e result viewers.
- O Autopsy apenas permite que um caso seja aberto de cada vez, para iniciar uma investigação.
- Fonte de dados é o termo usado para se referir a imagens de disco e arquivos lógicos que são adicionados a um caso.

[3] https://www.autopsy.com/download/

- Os resultados obtidos na fase de análise são geralmente afixados numa seção do Blackboard disponível no Autopsy. Esses artefactos analisados estão disponíveis de forma a serem visualizados pelo investigador.

No contexto de uma investigação, deve-se usar o fluxo de trabalho de análise de dados na ferramenta Autopsy [41], que se ilustra na figura 23.

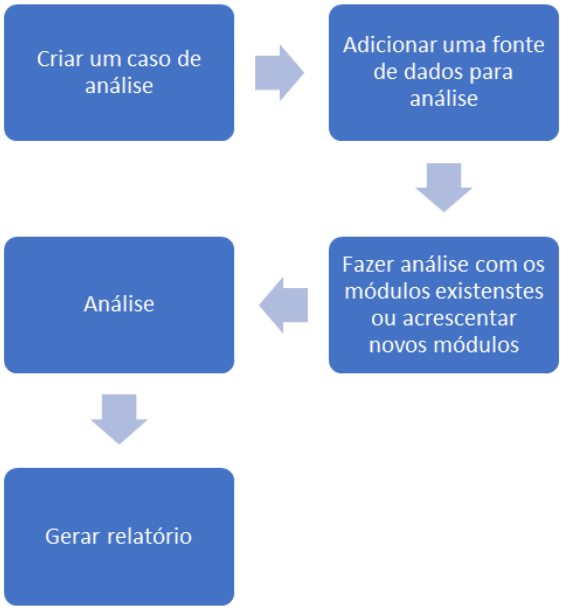


Figura 23 - Fluxo de trabalho na análise de dados

4. Análise forense de sistemas de infoentretenimento em veículos BMW

Este capítulo apresenta a arquitetura e funcionalidades dos sistemas IVI da marca BMW. Este capítulo mostra também os métodos usados para remover os sistemas IVI dos veículos, como recuperar informações nos sistemas IVI e quais os dados importantes para a análise forense.

4.1. Sistema IVI modelo CIC e NBT dos veículos BMW

Neste projeto foram analisados quatro sistemas IVI da marca BMW de veículos reais, dois do modelo Computer-In-Car (CIC) e dois do modelo Next Big Thing (NBT).

Os sistemas IVI modelo CIC HU HIGH que foram analisados neste projeto pertencem a um BMW do ano de 2012 e outro do ano 2010, ambos do modelo série 3. O sistema IVI modelo NBT EVO HU pertence a um BMW do ano 2017 modelo série 5 e série 7.

No geral, os dados recuperados dos sistemas IVI CIC e NBT podem ser úteis em certos tipos de investigações. A BMW introduziu funcionalidades nos sistemas IVI nos veículos por volta de 2001 com a primeira geração de BMW iDrive. Seguiu-se a próxima geração do sistema iDrive entre 2003 e 2008 e foi chamado de Car Communication Computer (CCC), posteriormente, o sistema IVI CIC. Este sistema IVI foi lançado em meados de 2008 e pode ser encontrado em alguns modelos da BMW até 2016. Atualmente estamos na geração do sistema NBT, que foi lançado em 2016 e pode ser encontrado até aos modelos presentes [42].

Os sistemas IVI da BMW são compostos por um display LCD no painel central, iDrive navigation que dá acesso ao Radio/CD/Multimédia e um controller iDrive, ou joystick entre os bancos da frente que é um controlador com teclas que dá acesso ao menu do iDrive navigation, como se ilustra na figura 24.

Figura 24 - Sistema IVI BMW [23]

Na figura 25 é exibido o display LCD central do sistema IVI dos modelos da marca BMW, que é conhecido como central information display (CID).

Figura 25 - Central information display sistema IVI [23]

A nível de arquitetura os sistemas IVI são compostos por hardware e software, possuem várias camadas, a camada do hardware, sistema operativo, Middleware e a camada de aplicação para finalmente chegar à camada HMI. [43]

Na figura 26 está ilustrado o diagrama do sistema IVI da BMW.

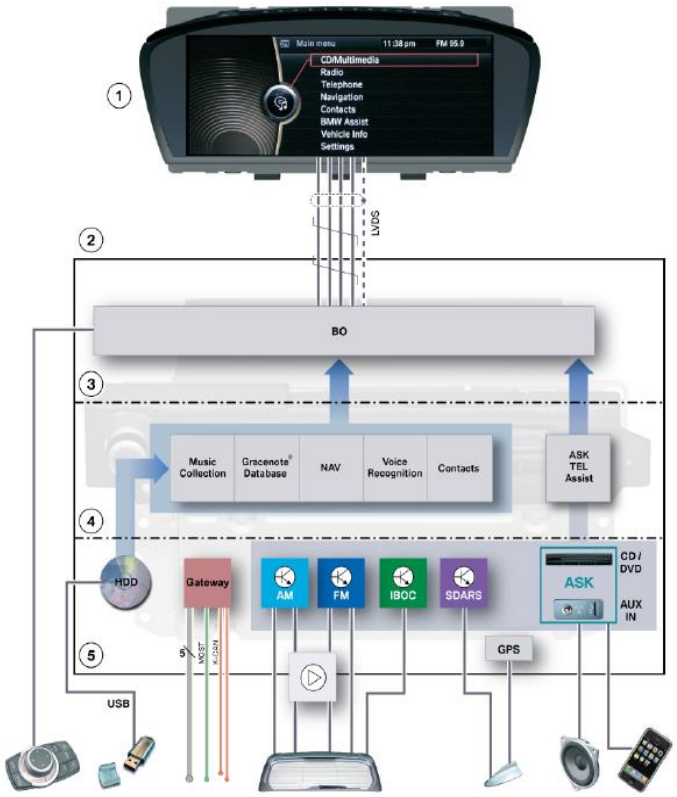


Figura 26 - Diagrama do sistema IVI BMW [44]

1. Central information display (CID).
2. Car information computer (CIC).
3. User interface (UI).
4. Aplicação de software.
5. Hardware.

A baixo estão mencionadas as camadas que constituem a arquitetura do sistema IVI da BMW.

- **Camada HMI**, exibição central de informações, é a principal interface do sistema IVI com a qual os utilizadores interagem com o equipamento.
- **Camada de aplicação,** é a camada que contém um conjunto de aplicações que fornecem vários recursos do sistema IVI, que permite realizar atividades ou tarefas para o benefício do utilizador, tais como, calendário, chamadas, Bluetooth, músicas, navegação GPS.
- **Camada de Middleware**, é uma camada que fica entre o sistema operativo e a camada de aplicação. Permite que as aplicações comuniquem com o Hardware e acedam aos recursos do sistema. Fornece vários componentes e serviços para realizar

todas as áreas funcionais da camada de aplicação, como Bluetooth, chamadas navegação GPS e áudio.

- **Camada do Sistema Operativo**, o sistema operativo faz a gestão de recursos de hardware e software. Fornece uma interface e permite que as aplicações sejam executadas.
- **Camada de Hardware**, esta camada são os componentes físicos do sistema IVI e é constituída por um CPU com todo o hardware necessário.

Estes sistemas IVI da BMW, são uma unidade "tudo em um", contém uma unidade de CD/DVD na parte frontal como ilustra a figura 27 e na parte traseira contém várias entradas como se ilustra na figura 28.

Figura 27 - Parte frontal sistema IVI

Figura 28 - Parte traseira sistema IVI

Este sistema IVI contém processadores, RAM e outros periféricos. Na figura 29 ilustra a memória RAM e o CPU. Na figura 30 ilustra o processador usado pelos sistemas IVI da BMW.

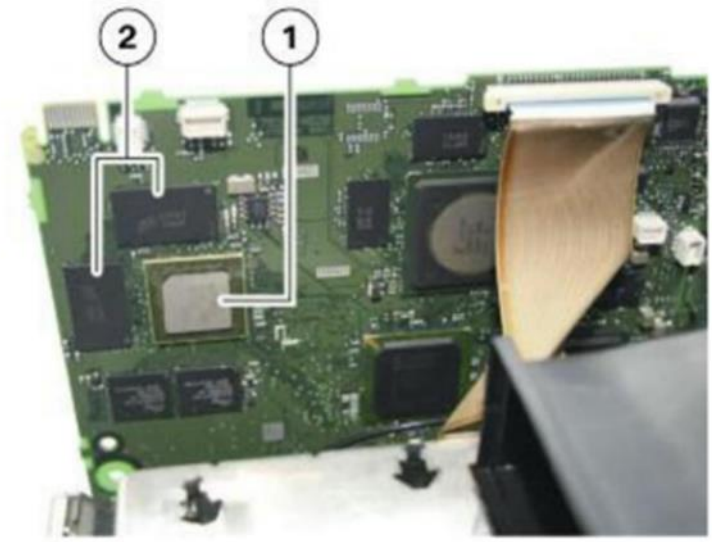


Figura 29 - Memória RAM e CPU sistemas IVI BMW

1. CPU.
2. Memória RAM.

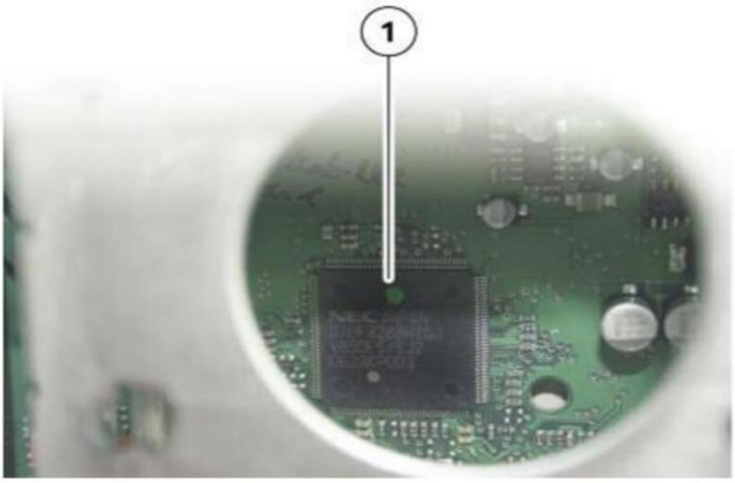


Figura 30 - Processador sistema IVI BMW

1. Processador.

No seu interior possui discos rígidos HDD. No modelo CIC o tamanho é de 80 GB e no modelo NBT é 200GB, como se ilustra nas figuras 31 e 32.

Figura 31 - Disco rígido HDD modelo NBT

Figura 32 - Disco rígido HDD modelo CIC

4.2.Funcionalidades do sistema IVI BMW

4.2.1. Navegação GPS

Permite obter a posição precisa do veículo, inserir um destino, uma rota, sendo estes dados guardados nos veículos. Sendo possível consultar os últimos destinos recentes, pontos de interesse, como consta na figura 33.

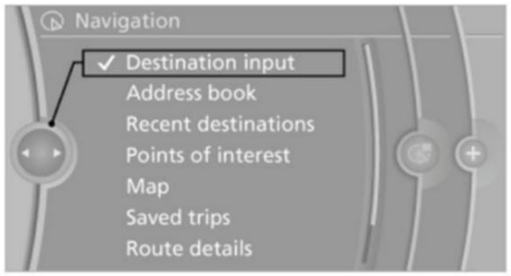


Figura 33 - Navegação sistema IVI [45]

4.2.2. Informações de trânsito

São informações de trânsito das estações de rádio que enviam um serviço de aviso de trânsito no Traffic Message Channel (TMC). São apresentadas no mapa e consideradas no cálculo do percurso. As informações sobre obstruções e perigos de trânsito são atualizadas constantemente. As informações de trânsito são exibidas no mapa por símbolos. As informações de trânsito são armazenadas numa lista. As diversas informações de trânsito são ordenadas por distância da posição atual do veículo.

4.2.3. Música

As músicas que constam nos CDs/DVDs e dispositivos ligados por USB, podem ser guardadas no próprio disco rígido do sistema IVI, como se ilustra na figura 34.

Figura 34 - Lista de músicas sistema IVI [45]

4.2.4. Dispositivos externos ligados ao sistema IVI

Vários dispositivos podem ser ligados ao veículo, conforme a figura 35.

Figura 35 – Dispositivos ligados ao sistema IVI [45]

4.2.5. Bluetooth

Permite sincronizar o telemóvel com o veículo. Antes de passar os contactos e histórico de chamadas para o sistema IVI do veículo recebe um alerta no telefone para dar autorização de acesso, como ilustra na figura 36. Após a sincronização bem-sucedida, os dados do telemóvel, contactos e históricos de chamadas são transferidos para os sistemas IVI. O Bluetooth também tem a função de fazer chamadas de alta-voz.

Figura 36 - Pedido de acesso aos contatos e histórico de chamadas

4.2.6. Contatos

A lista telefónica do telemóvel é transferida para o veículo após a sua deteção. Requer sempre confirmação do utilizador. O utilizador tem opção de apagar no próprio veículo os contactos.

Os contatos estão guardados numa base de dados e no próprio sistema IVI, permite fazer a gestão dos nomes, endereços e números de telefone, como se ilustra na figura 37.

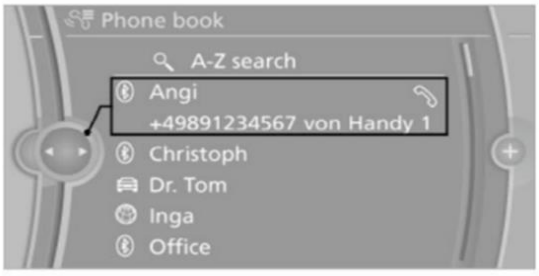


Figura 37 - Phonebook [45]

4.2.7. Mensagens / e-mail

Permite visualizar mensagens de texto e e-mails, como ilustra na figura 38.

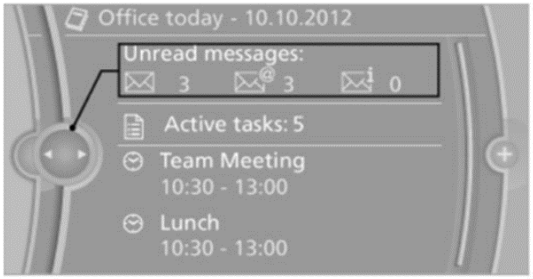


Figura 38 - Mensagens [45]

4.2.8. Calendário/Tarefas

Permite a exibição de compromissos que constam no calendário e selecionar as tarefas que devem ser transferidas de um telemóvel, como ilustra nas figuras 39 e 40.

Figura 39 - Calendário [45]

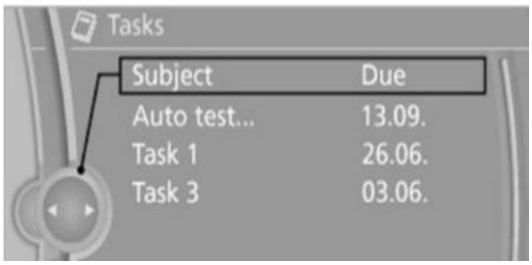


Figura 40 - Tarefas [45]

4.2.9. BMW Live

É um serviço online que disponibiliza alguns serviços, por exemplo, informações sobre as previsões meteorológicas e a localização atual.

4.2.10. APP BMW ConnectedDrive

Através desta app obtemos o controlo remoto do veículo, que pode ser para controlo das portas, ventilação interior, localização do veículo. Permite ainda uma visão geral de dados do veículo, referente ao combustível, quilómetros e a próxima manutenção. Possibilita ainda fazer marcações de manutenção e visualizar o n.º do VIN, como consta nas figuras 41, 42 e 43.

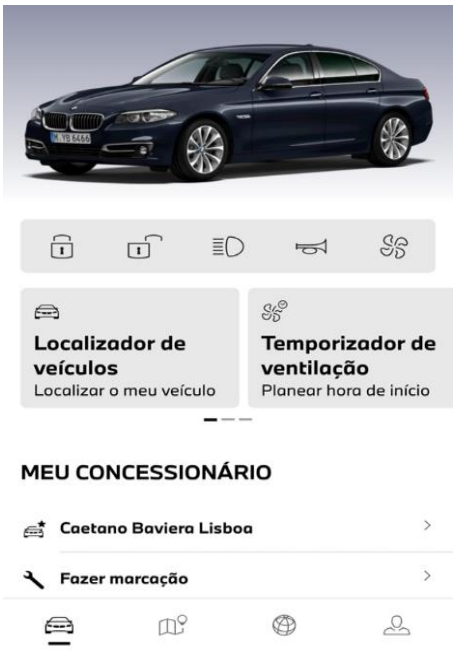


Figura 41 - Visão geral da APP BMW ConnectedDrive

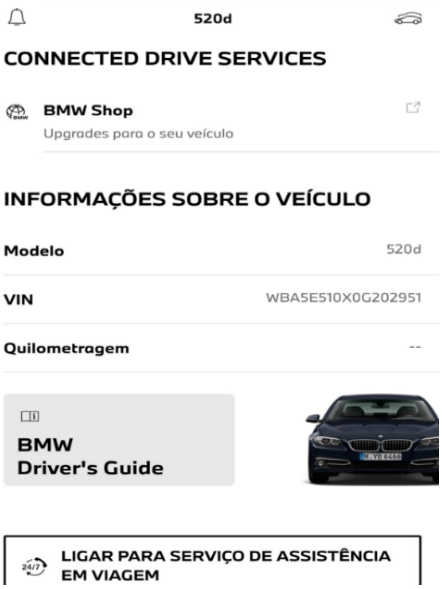


Figura 42 - Informações do Veículo

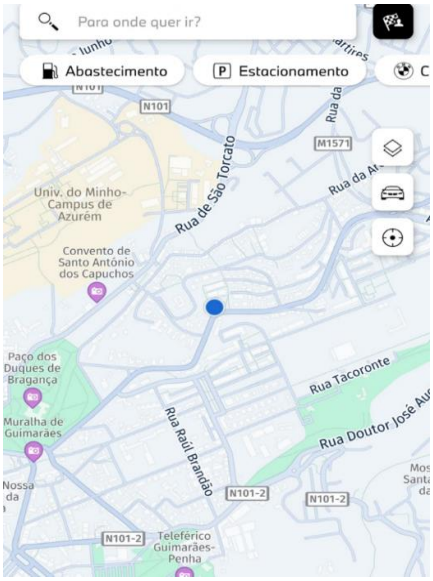


Figura 43 - Localização do veículo

4.2.11. Apple Car play/Android Auto

Permite que certas funções de um iPhone ou do Android sejam usadas no sistema IVI através da ligação por Bluetooth ou por cabo USB.

4.2.12. Comparação das funcionalidades do modelo CIC e NBT

Estes dois modelos apresentam bastantes diferenças. Da análise efetuada podemos verificar que o modelo NBT é mais completo e tem mais funcionalidades. O modelo NBT é mais recente que o modelo CIC. Na tabela 1 é mencionada a comparação entre as várias funcionalidades do modelo CIC e NBT.

Tabela 1 - Comparação das funcionalidades entre sistema IVI CIC e NBT

Funcionalidades	NBT EVO HU	CIC HU HIGH
Contactos	☑	☑
Mensagens	☑	
E-mail	☑	
Chamadas	☑	
Bluetooth	☑	☑
Internet	☑	
Modelo do smartphone	☑	
Marca do smartphone	☑	
Dispositivos ligados	☑	☑
Navegação/GPS	☑	☑
Histórico WEB	☑	
Cookies	☑	
Álbuns de música	☑	☑
Cantores musicais	☑	☑
APP BMW ConnectedDrive	☑	
Apple carplay/Android auto	☑	
Calendário/Tarefas	☑	☑
Traffic Message Channel (TMC)	☑	
Rádio	☑	☑

4.3. Remoção dos sistemas IVI do veículo

Na remoção aos sistemas IVI deve-se cumprir os seguintes procedimentos [46] [47]:

1. Os equipamentos a ser analisados, devem ser manuseados de forma adequada para evitar modificações ou destruição de dados;
2. Devemos efetuar a apreensão do veículo ou equipamento do sistema IVI, tendo atenção que devemos fazer sempre a custódia da prova;

3. Nesta fase, antes de qualquer intervenção devemos saber qual a tecnologia que vamos analisar e como funciona, em virtude de existirem várias marcas, vários modelos e serem todos diferentes;
4. Antes de remover qualquer sistema IVI, devemos escolher a melhor abordagem e efetuar uma inspeção ocular;
5. Desligar o veículo, retirar as chaves e desligar os bornes da bateria;
6. Esperar mais ou menos 15 minutos depois de desligar tudo;
7. Verificar e certificar se o painel de instrumentos e as luzes estão desligadas;
8. Isolar e desligar o veículo das redes Wi-Fi, Bluetooth e de ligações com outros dispositivos, nomeadamente telefones, para evitar que ocorram interações indesejadas e que danifiquem ou adulterem a prova;
9. Todos os processos de recolha de prova devem ser documentados, desde a hora e a data em que essas etapas são executadas;
10. Deve anotar as diferenças entre a hora atual no país em que está a fazer análise e a hora exibida no sistema IVI;
11. Documentar e fotografar o exterior e o interior do veículo, antes da remoção dos sistemas IVI;
12. Fazer anotações relativas ao veículo, nomeadamente as condições e o estado em que encontramos o veículo;
13. Retirar o número Vehicle Identification Number (VIN), que fornece informações sobre a marca, modelo e ano do veículo;
14. Fotografar ao pormenor o sistema IVI, antes e depois de o retirar, se estiver ligado tirar uma foto ao ecrã;
15. Por fim remover o sistema IVI do veículo;
16. Fazer o transporte do equipamento de forma adequada, protegido de quedas, choques e temperaturas extremas [6];
17. Por fim desmontar o sistema IVI, para aceder ao disco rígido.

É de realçar que neste projeto não foi efetuada nenhuma remoção do sistema IVI, foi possível adquirir quatro sistemas IVI já removidos do interior dos veículos. Neste projeto foram adquiridos dois sistemas IVI CIC do modelo série 3 do ano de 2010, 2012 e um sistema IVI NBT do modelo série 5 e série 7 do ano 2017. O processo de remoção é composto por vários passos, a seguir é descrito um guia como se removem os sistemas IVI referentes aos modelos

que foram analisados neste projeto, esta informação é baseada na aplicação da Berla iVe Mobile para android[4].

4.3.1. Remoção do sistema IVI NBT da marca BMW modelos serie 5 e 7

Remoção das duas peças de guarnição ao longo do chão, conforme figura 44 e 45.

Figura 44 - Remoção da guarnição ao longo do chão [4]

Figura 45 - Remoção da guarnição [4]

[4] https://play.google.com/store/apps/details?id=com.berla.ivemobile&hl=en

Remove-se a peça do tablier do lado direito, entre a porta lateral do passageiro e porta luvas. Ao remover, deve-se ter cuidado para não afetar o airbag conforme na figura 46.

Figura 46 - Remoção da peça do lado direito do tablier [4]

Remoção dos parafusos do lado direito do tablier entre a porta lateral do passageiro e porta luvas, conforme na figura 47.

Figura 47 - Remoção o parafuso na extremidade do tablier [4]

Remoção da peça que se encontra ao longo do tablier e por cima do porta-luvas. Esta peça inclui as aberturas de ar. Conforme na figura 48.

Figura 48 - Remoção da guarnição que corre ao longo do tablier [4]

Nesta fase retiramos a parte frontal do sistema IVI, conforme figura 49.

Figura 49 - Remoção frontal do sistema IVI [4]

Depois da remoção da parte frontal, acedemos à unidade do sistema IVI, desaperta-se os parafusos para o remover, conforme figura 50. A seguir desligamos todos os cabos que estão ligados ao sistema IVI, por fim removemos os parafusos da parte superior e frontal ao lado da parte preta (leitor CD/DVD), conforme figura 51 e 52.

Figura 50 - Remoção do sistema IVI do veículo [4]

Figura 51 - Remoção dos parafusos leitor CD/DVD [4]

Figura 52 - Remoção dos parafusos da parte de cima [4]

Removemos a parte superior do sistema IVI, acedemos à unidade de CD/DVD, a seguir levantamos cuidadosamente a unidade de CD/DVD para retirar o disco rígido HDD. Devemos ter cuidado para não danificar os cabos que ligam à unidade CD/DVD, e por fim retirar o disco rígido. Para retirar o disco rígido, devemos empurrar o disco rígido para a parte de trás para o desprender, conforme figura 53, 54 e 55.

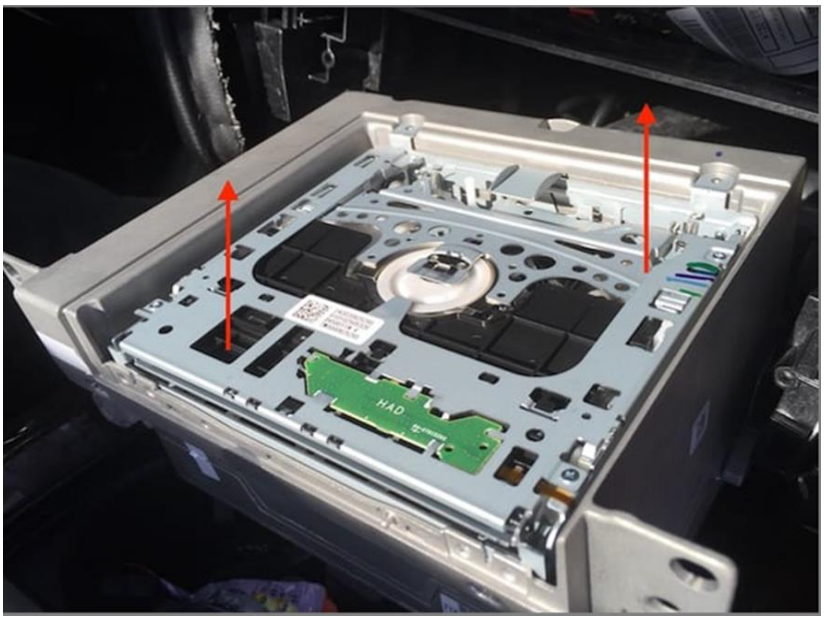


Figura 53 - Remoção da unidade CD/DVD [4]

Figura 54 - Sistema IVI sem a unidade CD/DVD [4]

Figura 55 – Remoção do disco rígido HDD [4]

Na figura 56, ilustra o disco rígido que foi removido dos sistemas IVI NBT.

Figura 56 - Disco rígido do sistema IVI NBT

4.3.2. Remoção do sistema IVI CIC da marca BMW modelos serie 3.

Remoção da peça do tablier, que fica por baixo da primeira entrada de ventilação do lado esquerdo, conforme figura 57 e 58.

Figura 57 - Remoção da peça do Tablier parte1 [4]

Figura 58 - Remoção da peça do tablier parte 2 [4]

Removemos cuidadosamente a peça do tablier, pedaço estreito que corre ao longo do painel e acima do porta-luvas. Esta peça inclui as aberturas de ar, conforme figura 59.

Figura 59 - Remoção da parte de ventilação [4]

Depois de retirar a parte de ventilação, desapertamos os parafusos para retirar a parte frontal do sistema IVI, como se ilustra na figura 60.

Figura 60 - Remoção da parte frontal do sistema IVI [4]

Depois de retirar a parte frontal acedemos ao sistema IVI, desapertamos os parafusos, como se ilustra na figura 61.

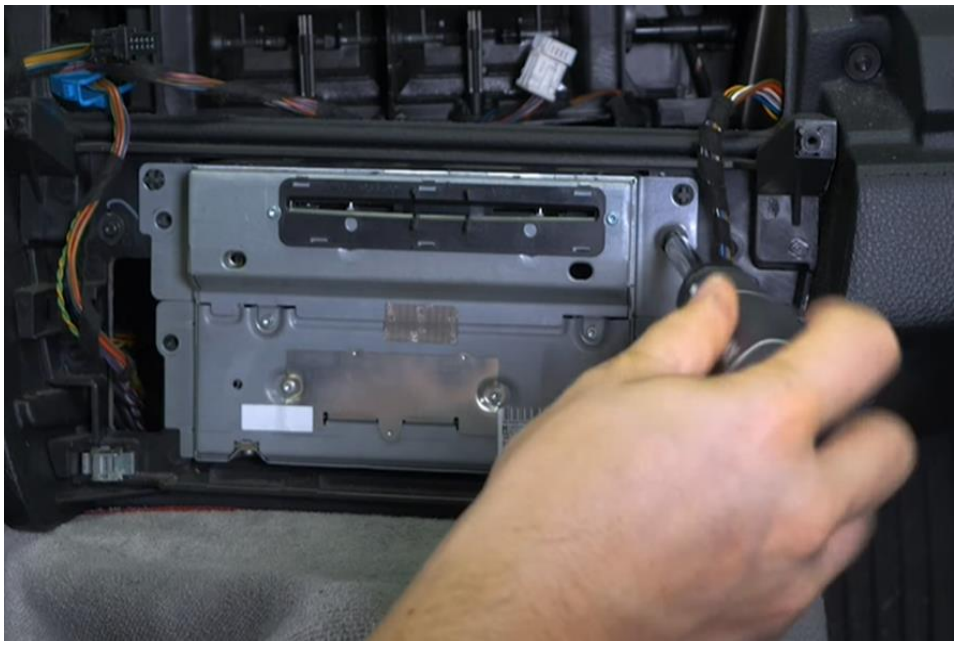


Figura 61 - Remoção do sistema IVI [4]

Para retirar o disco rígido que se encontra no interior do sistema IVI, removemos a tampa frontal que se encontra por baixo da entrada do CD/DVD, como se ilustra na figura 62.

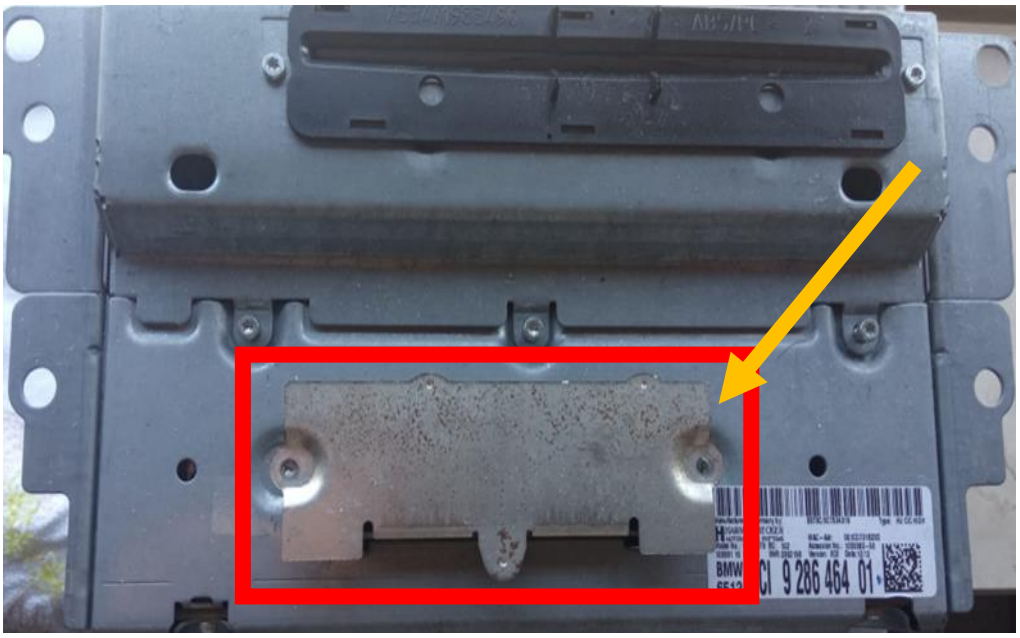


Figura 62 - Remoção do disco rígido

A figura 63 e 64, demonstra os discos rígidos que foram removidos dos sistemas IVI CIC.

Figura 63 - Disco rígido do sistema IVI CIC ano 2010

Figura 64 - Disco rígido do sistema IVI CIC ano 2012

4.4.Aquisição física/cópia forense

Para este projeto foi possível adquirir fisicamente três discos rígidos de três sistemas IVI diferentes de veículos reais. Foi utilizada a ferramenta forense FTK Imager, versão 4.2.0.13, para efetuar uma aquisição física forense dos três discos rígidos, que estão mencionados nas tabelas 2 e 3.

Tabela 2 - Características dos discos rígidos do modelo CIC

Tag ID	Marca	Modelo	N-º Série	interface	capacidade	Tipo
CIC2012serie3	TOSHIBA	TOSHIBA MK8050GAC	22F2T1P8T	ATA/IDE	80GB	HDD
CIC2010serie3	TOSHIBA	TOSHIBA MK8050GAC	98I8T4K8T	ATA/IDE	80GB	HDD

Tabela 3 - Características dos discos rígidos do modelo NBT

Tag ID	Marca	Modelo	N-º Série	interface	capacidade	Tipo
NBT2017serie5	HGST	HEJ423220H 9E300	44JVJ1UP	SATA	200GB	HDD

O sistema IVI do modelo NBT série 7 do ano 2017, não foi efetuada nenhuma imagem física ao disco rígido, em virtude de terem sido já fornecidas as partições já montadas do disco HDD desse veículo.

4.4.1. AccessData Ftk Imager

Para criar uma imagem do disco rígido, clicamos em File e depois em Create Disk Image, como demonstra na figura 65.

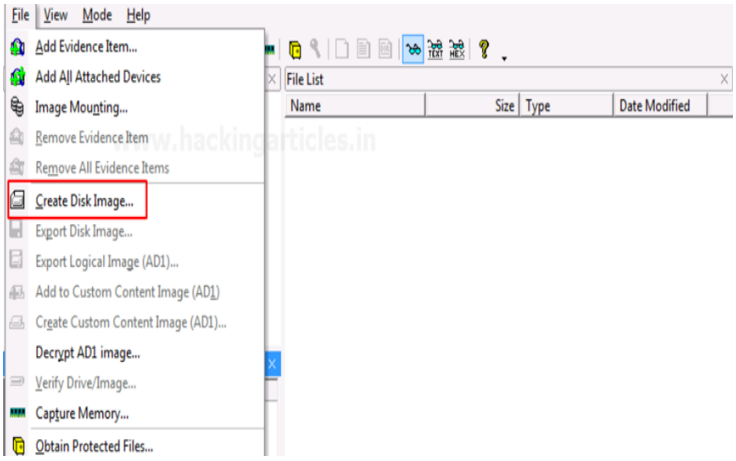


Figura 65 - Criar imagem

A seguir selecionamos o tipo de evidência, como se ilustra na figura 66, neste projeto foi efetuada imagem física.

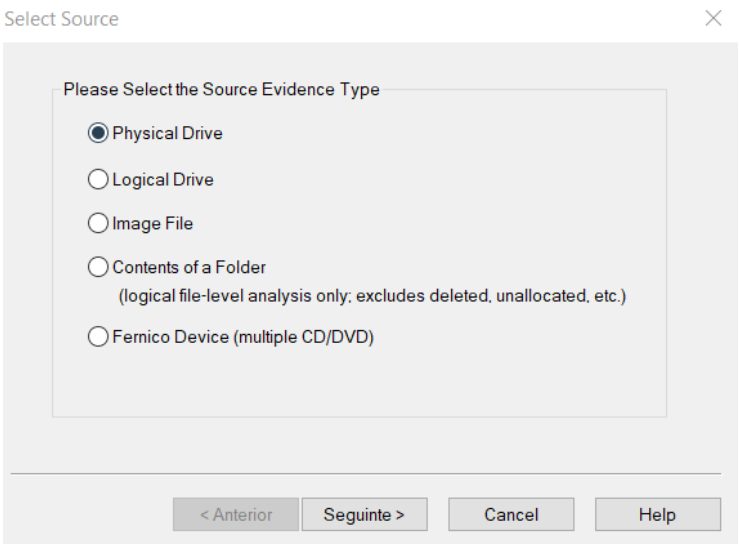


Figura 66 - Tipos de análise

Na figura 67, selecionamos a unidade de armazenamento da qual queremos fazer a imagem.

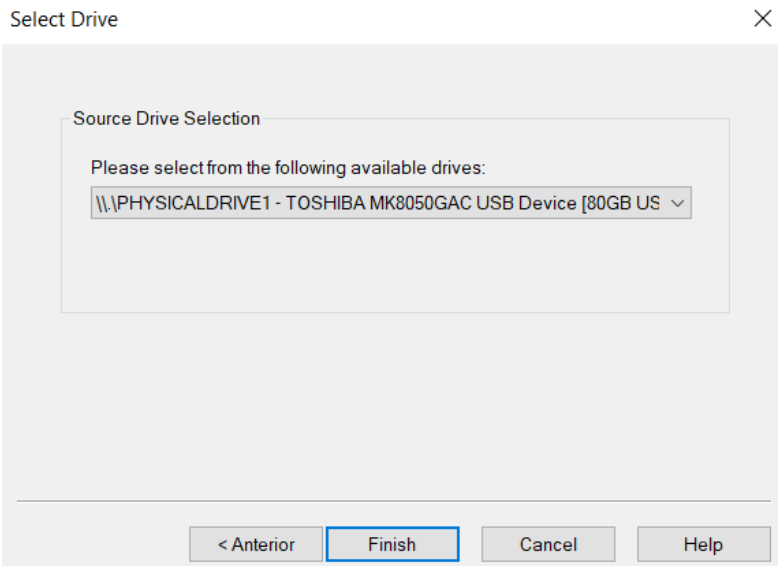


Figura 67 – Selecionar o tipo de evidência para análise

Na figura 68, ilustra o destino da imagem que será criada. Selecionamos o botão Add, que vai permitir selecionar o formato da imagem que desejamos criar.

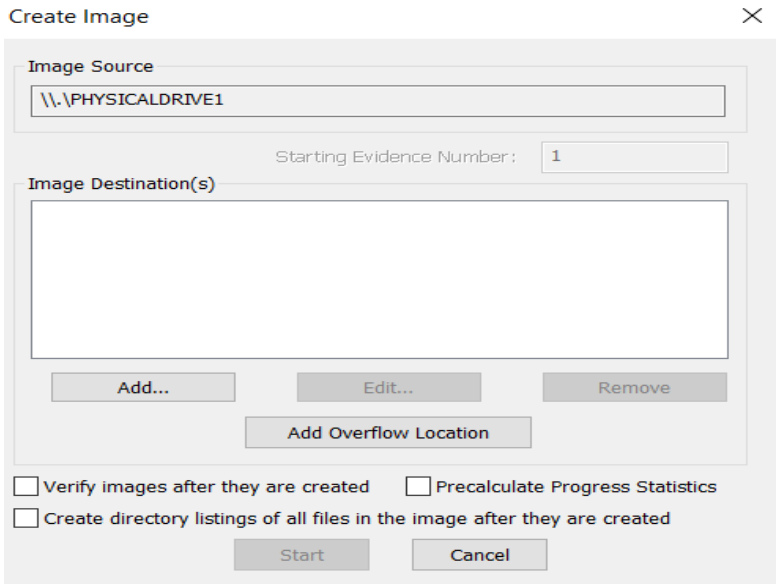


Figura 68 - Escolher o local para guardar a imagem

Depois de selecionarmos o botão Add, vamos selecionar o formato da imagem que desejamos criar, como consta na figura 69. Neste projeto foi usado o formato Raw (dd), este formato permite efetuar uma cópia bit a bit da prova original, que é criada sem quaisquer

acréscimos ou exclusões, a imagem resultante é uma replica exata do dispositivo de armazenamento original, incluindo todos os dados, metadados e estruturas do sistema de ficheiros.

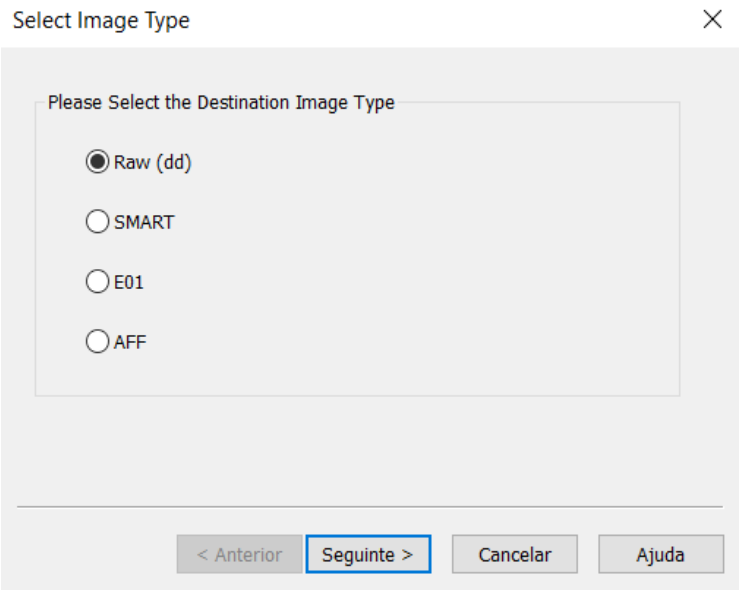


Figura 69 - Formatos para criar imagens

A figura 70 ilustra como permite adicionar detalhes, o número do caso, o nome do examinador e colocar notas.

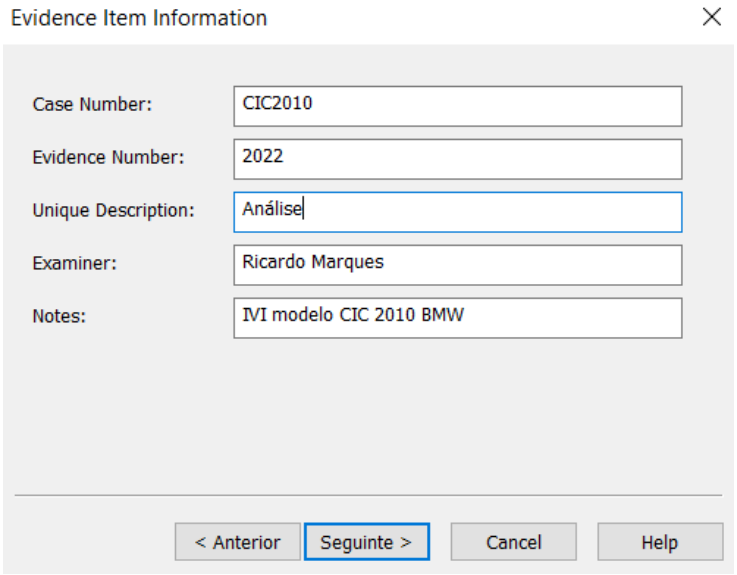


Figura 70 – Adicionar informação ao caso em análise

Finalmente, adicionamos o destino do arquivo da imagem e colocamos um nome à imagem que vai ser criada, como ilustra na figura 71. Na opção “Image Fragment Size”, foi utilizado o tamanho zero (0), esta configuração permite que a imagem seja criada como um único ficheiro completo, em vez de ser dividida em fragmentos menores.

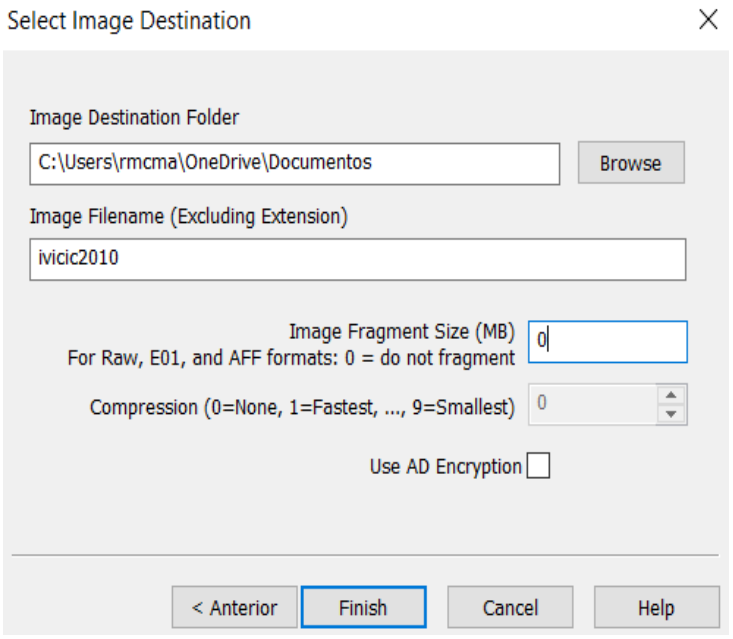


Figura 71 - Selecionar o destino da imagem

Na figura 72, ilustra o início do processo da criação da imagem ao selecionarmos Start.

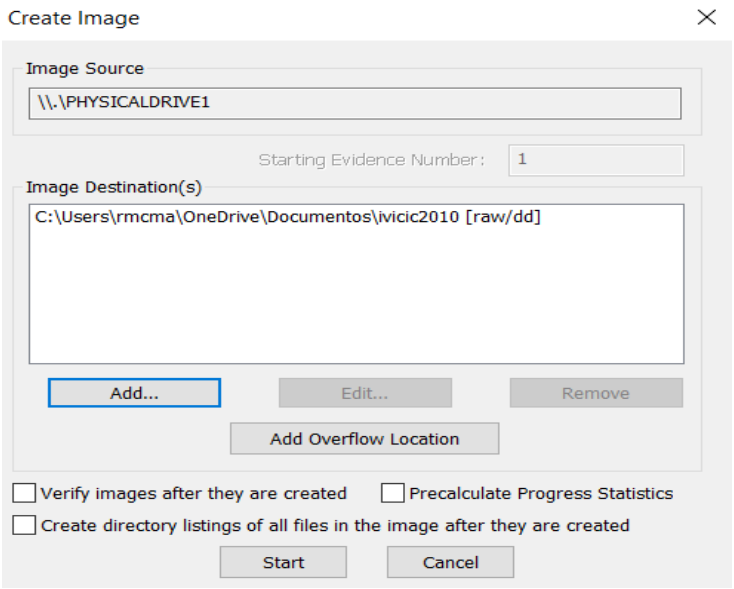


Figura 72 - Inicio do processo

4.5.Montar as Partições

Como a ferramenta Autopsy não suporta o sistema de ficheiros QNX, para isso foi necessário montar as partições manualmente através de uma distribuição Linux e a seguir carregar manualmente essas partições para o Autopsy para efetuar a análise.

Depois de criar as imagens dos discos rígidos dos vários sistemas IVI, começamos por um processo de análise manual, para isso, foi usado Kali Linux para montar as partições [48].

A seguir vamos explicar a montagem das partições, primeiro usamos o comando *fdisk*, que permite listar todas as partições disponíveis.

sudo fdisk -lu test.dd

O comando **"sudo fdisk -lu test.dd"** é usado para listar informações sobre as partições presentes no ficheiro de imagem de disco "test.dd", que corresponde ao disco rígido do sistema IVI NBT do modelo série 5 do ano 2017.

A seguir está uma explicação de cada opção do comando:

- **"sudo":** Executa o comando com privilégios administrativos.
- **"fdisk":** Comando utilizado para efetuar a gestão das partições num disco.
- **"-l":** Opção usada para listar informações sobre as partições em todos os dispositivos detetados no sistema, incluindo dispositivos de bloco e arquivos de imagem de disco.
- **"-u":** Opção usada para exibir a tabela de partições em unidades de setores.

A figura 73, apresenta um total de 4 partições, neste caso o tipo de ficheiros é desconhecido. Foi usado o Comando *binwalk test.dd* que permite pesquisar os setores de inicialização do sistema operativo QNX, que se destinam a fornecer informações sobre os sistemas de ficheiros presentes, neste caso sabemos que possui sistema de ficheiros QNX4.

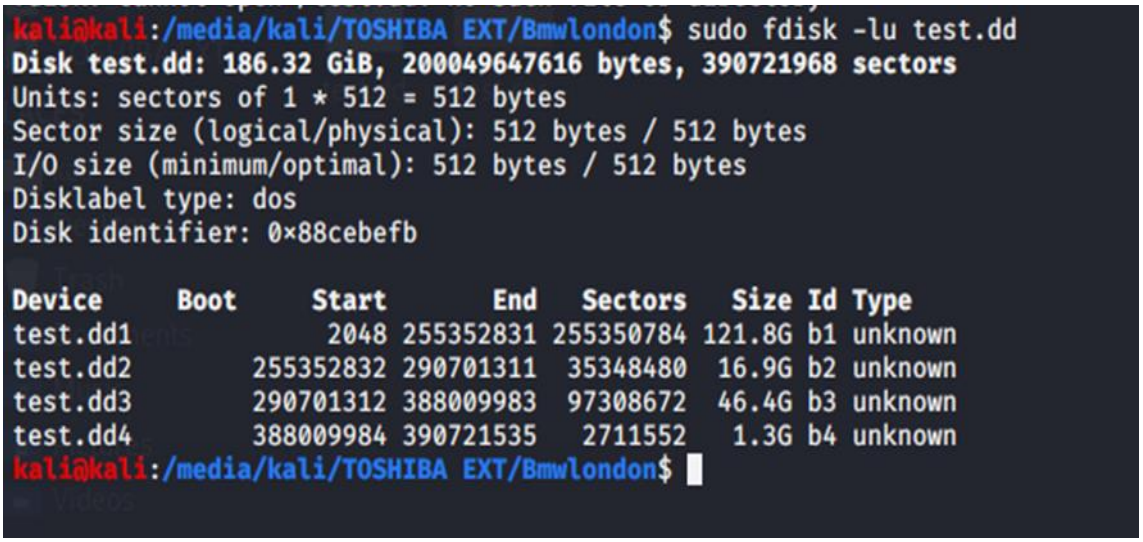


Figura 73 - Partições do sistema IVI NBT 2017 serie 5

sudo fdisk -lu simao.001

Executamos o comando fdisk à imagem simao.001, que corresponde ao disco rígido do sistema IVI CIC do modelo série 3, ano 2012. A figura 74, apresenta um total de 7 partições e o tipo de sistema de ficheiros é QNX4.

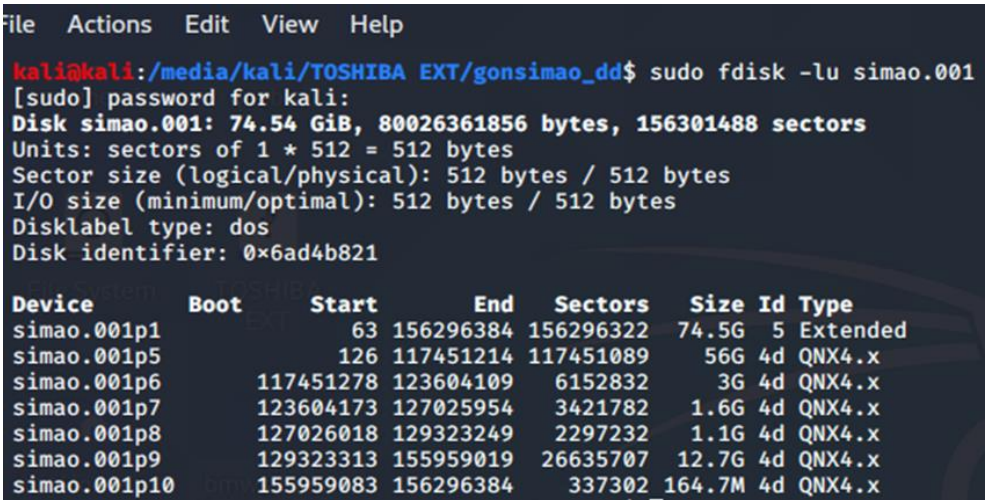


Figura 74 - Partições do sistema IVI CIC 2012 serie 3

sudo fdisk -lu hendo.001

Executamos o comando fdisk à imagem hendo.001, que corresponde ao disco rígido do sistema IVI CIC do modelo série 3, ano 2010. A figura 75, apresenta um total de 7 partições e o tipo de sistema de ficheiros é QNX4.

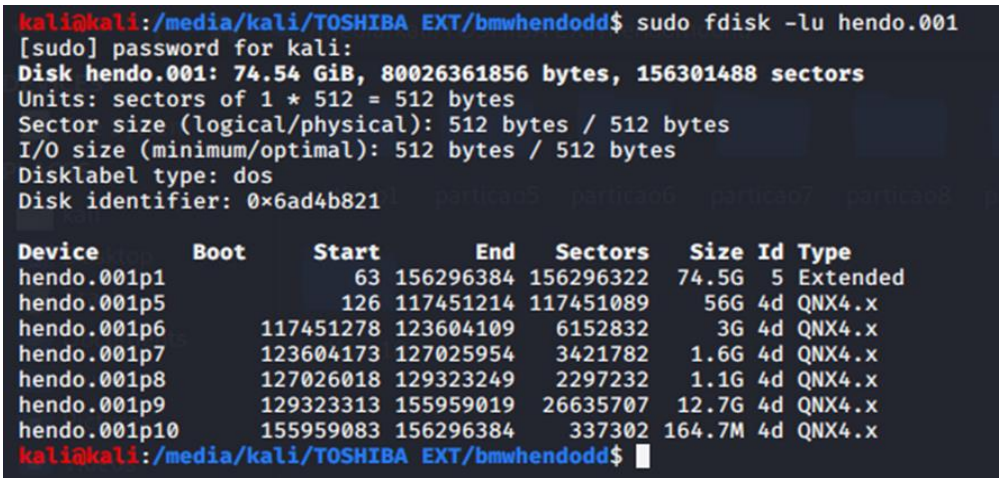


Figura 75 - Partições do sistema IVI CIC 2010 serie 3

A seguir usamos o comando "**sudo losetup --partscan --find –show**", como se ilustra na figura 76, 77 e 78. Este comando é usado para criar um dispositivo loopback para um arquivo de imagem de disco chamado "test.dd" e exibir o caminho para o dispositivo criado.

A seguir está uma explicação de cada opção do comando:

- **"sudo":** Executa o comando com privilégios administrativos.
- **"losetup":** Comando utilizado para configurar loop devices no sistema.
- **"--partscan":** Analisa a tabela de partições do ficheiro e cria um dispositivo loopback para cada partição encontrada.
- **"--find":** Encontra o próximo dispositivo loopback livre para ser utilizado.
- **"--show":** Exibe o caminho para o dispositivo loopback criado.

O comando **"sudo ls -la /dev/loop0*"** é usado para listar todas as informações de permissão e propriedade dos dispositivos loopback disponíveis no sistema que começam com **"loop0".**

A seguir está uma explicação de cada opção do comando:

- **"sudo":** Executa o comando com privilégios administrativos.
- **"ls":** Comando utilizado para listar informações sobre ficheiros e diretórias.
- **"-la":** Opção usada para listar todas as informações de permissão e propriedade dos ficheiros, incluindo ficheiros ocultos.
- **"/dev/loop0*":** Especifica o padrão de nomes de ficheiros a serem listados. Nesse caso, todos os ficheiros que começam com "loop0" na diretoria "/dev/" serão listados.

sudo losetup --partscan --find --show test.dd
sudo ls -la /dev/loop0*

Figura 76 - Integração das partições num dispositivo lógico sistema IVI NBT 2017

sudo losetup --partscan --find --show simao.001
ls -la /dev/loop0*

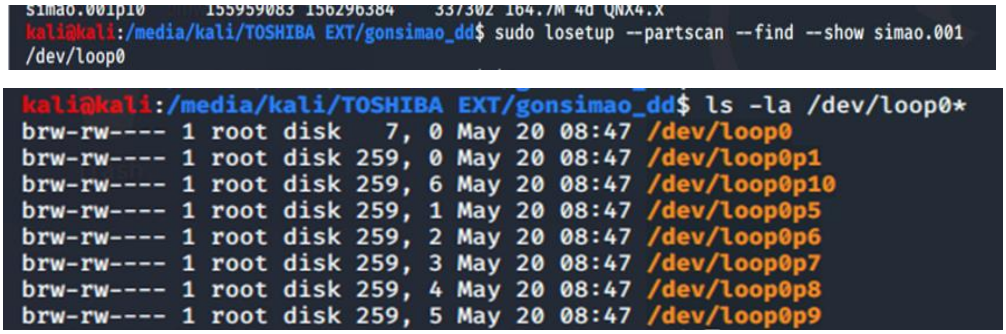


Figura 77 - Integração das partições num dispositivo lógico sistema IVI CIC 2012

sudo losetup --partscan --find --show hendo.001
ls -la /dev/loop0*

Figura 78 - Integração das partições num dispositivo lógico sistema CIC 2010

Inicialmente foi usado o sistema de ficheiros QNX4, que especifica o sistema de ficheiros a ser montado, no entanto, a montagem não foi possível, deu erro. Isto deve-se à

incompatibilidade, o sistema de ficheiros QNX4 é antigo e não é suportado. Neste caso foi usado o sistema de ficheiros QNX6, que é o sistema de ficheiros atual e usado pela QNX Software Systems. Foi usado o comando **"sudo mount -r -t qnx6 /dev/loop0p1 /home/kali/Desktop/bmwsimao/particao1/"** é usado para montar um sistema de ficheiros somente de leitura do tipo **"qnx6"** a partir do dispositivo **"/dev/loop0p1"** na diretória **"/home/kali/Desktop/bmwsimao/particao1/"**.

A seguir é detalhado cada comando e suas opções:

- **"sudo":** Este comando é usado para executar o comando subsequente com privilégios administrativos.
- **"mount":** Este comando é usado para montar um sistema de ficheiros numa diretoria do sistema de ficheiros Linux.
- **"-r":** Esta opção é usada para montar o sistema de ficheiros em modo somente de leitura, o que significa que não podemos modificar os arquivos no sistema de ficheiros.
- **"-t qnx6":** Esta opção especifica o tipo de sistema de ficheiros a ser montado, neste caso, é "QNX6".
- **"/dev/loop0p1":** Este é o ficheiro de dispositivo que representa a partição do sistema de ficheiros a ser montado. "/dev/loop0" representa um dispositivo de loopback e "p1" representa a primeira partição nesse dispositivo.
- **"/home/kali/Desktop/bmwsimao/particao1/":** Esta é a diretoria onde o sistema de ficheiros será montado.

Como se ilustra nas figuras 79 e 80, para montar as partições do sistema IVI CIC foram usados os seguintes comandos:

sudo mount -r -t qnx6 /dev/loop0p5 /home/kali/Desktop/bmwsimao/particao5/

sudo mount -r -t qnx6 /dev/loop0p6 /home/kali/Desktop/bmwsimao/particao6/

sudo mount -r -t qnx6 /dev/loop0p7 /home/kali/Desktop/bmwsimao/particao7/

sudo mount -r -t qnx6 /dev/loop0p8 /home/kali/Desktop/bmwsimao/particao8/

sudo mount -r -t qnx6 /dev/loop0p9 /home/kali/Desktop/bmwsimao/particao9/

sudo mount -r -t qnx6 /dev/loop0p10 /home/kali/Desktop/bmwsimao/particao10/

sudo mount -r -t qnx6 /dev/loop0p1 /home/kali/Desktop/bmwsimao/particao1/

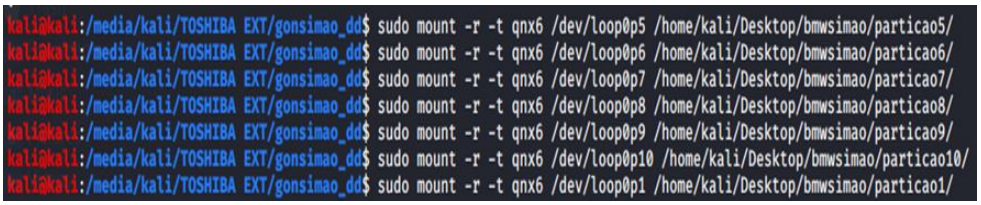


Figura 79 - Montagem das partições

sudo mount -r -t qnx6 /dev/loop0p5 /home/kali/Desktop/bmwsimao/particao5/

sudo mount -r -t qnx6 /dev/loop0p6 /home/kali/Desktop/bmwhendo/particao6/

sudo mount -r -t qnx6 /dev/loop0p7 /home/kali/Desktop/bmwhendo/particao7/

sudo mount -r -t qnx6 /dev/loop0p8 /home/kali/Desktop/bmwhendo/particao8/

sudo mount -r -t qnx6 /dev/loop0p9 /home/kali/Desktop/bmwhendo/particao9/

sudo mount -r -t qnx6 /dev/loop0p10 /home/kali/Desktop/bmwhendo/particao10/

sudo mount -r -t qnx6 /dev/loop0p1 /home/kali/Desktop/bmwhendo/particao1

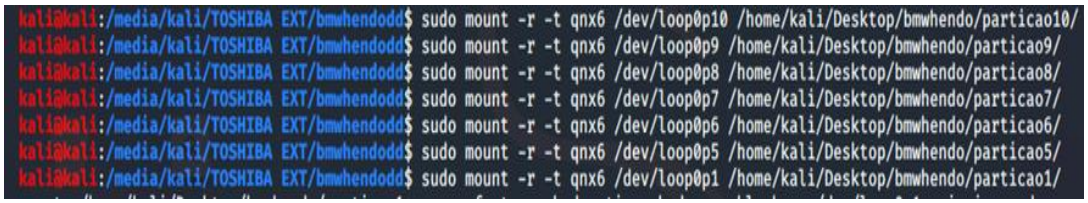


Figura 80 - Montagem das partições

Como se ilustra na figura 81, para montar as partições do sistema IVI NBT foram usados os seguintes comandos:

sudo mount -t qnx6 /dev/loop3p1 /dev/disk

sudo mount -t qnx6 /dev/loop3p2 /dev/disk

sudo mount -t qnx6 /dev/loop3p3 /dev/disk

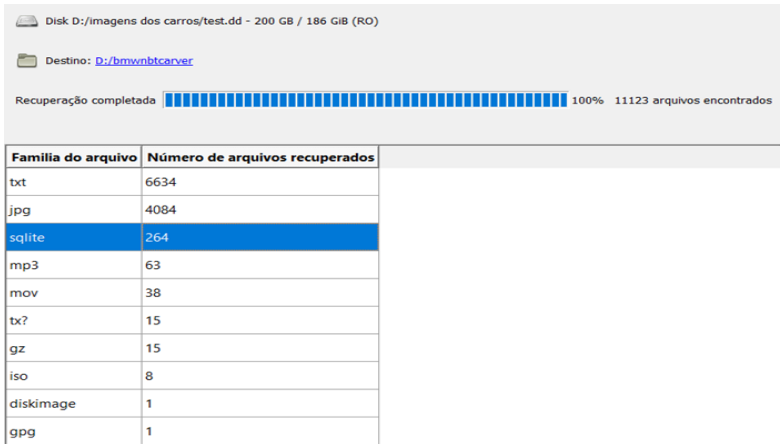
sudo mount -t qnx6 /dev/loop3p4 /dev/disk

```
kali@kali:/media/kali/TOSHIBA EXT/Bmwlondon$ sudo mount -t qnx6 /dev/loop3p1 /dev/disk
kali@kali:/media/kali/TOSHIBA EXT/Bmwlondon$ sudo mount -t qnx6 /dev/loop3p2 /dev/disk
kali@kali:/media/kali/TOSHIBA EXT/Bmwlondon$ sudo mount -t qnx6 /dev/loop3p3 /dev/disk
kali@kali:/media/kali/TOSHIBA EXT/Bmwlondon$ sudo mount -t qnx6 /dev/loop3p4 /dev/disk
kali@kali:/media/kali/TOSHIBA EXT/Bmwlondon$
```

Figura 81 - Montagem das partições

4.6. Qphotorec

Antes de proceder à análise no Autopsy, vamos fazer uma análise à imagem criada pelo Ftk Imager, na ferramenta Qphotorec. Como o Autopsy não permite utilizar photorec carver, por não reconhecer o sistema de ficheiros QNX, como alternativa, usamos o Qphotorec para recuperar dados excluídos, como podemos visualizar no caso do sistema IVI NBT. A figura 82 mostra a quantidade total de dados que foram encontrados, neste caso foram 11123 ficheiros.



Família do arquivo	Número de arquivos recuperados
txt	6634
jpg	4084
sqlite	264
mp3	63
mov	38
tx?	15
gz	15
iso	8
diskimage	1
gpg	1

Figura 82 -Recuperação de ficheiros

Depois da ferramenta Qphotorec efetuar a análise, foram recuperados vários ficheiros apagados. No meio de tantos ficheiros existem ficheiros importantes, no caso da análise do sistema IVI NBT série 5 do ano 2017, foram encontradas 4 bases de dados SQLite, f287105230, f287269072, f287629400 e f288251944, estas bases de dados não estavam nas partições que foram montadas. Estas bases de dados são importantes porque têm mensagens trocadas dos telefones que se sincronizaram ao sistema IVI.

4.7.Organização interna do sistema IVI BMW NBT

A organização interna do sistema IVI modelo NBT, é constituída por 4 partições, a partição 1 está relacionada com a parte de navegação GPS, como se ilustra na figura 83.

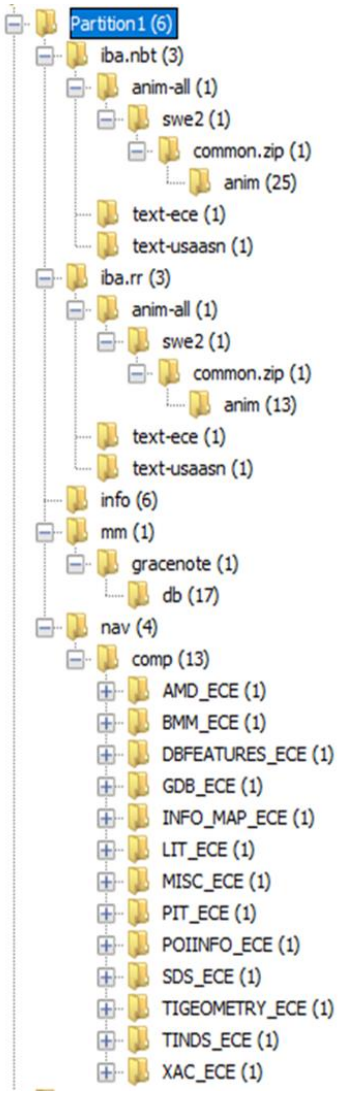


Figura 83 - Partição 1 modelo NBT

A partição 2 está relacionada com a parte de contactos telefónicos, chamadas, SMS, e-mail, tarefas, calendário, browser, cookies, dados do Bluetooth, marca e modelo do telefone, conforme apresenta a figura 84.

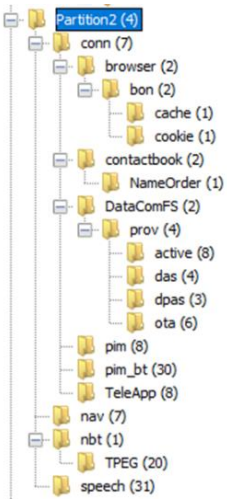


Figura 84 - Partição 2 modelo NBT

A partição 3 está relacionada com o áudio e vídeo, como se ilustra na figura 85.

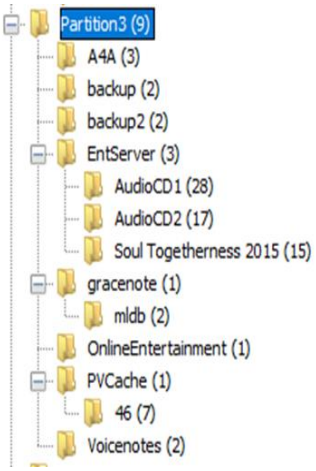


Figura 85 - Partição 3 modelo NBT

A partição 4 está relacionada com o sistema operativo, como se ilustra na figura 86.

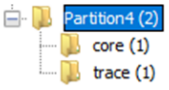


Figura 86 - Partição 4 modelo NBT

4.8. Organização interna do sistema IVI BMW CIC

A organização interna do sistema IVI modelo CIC, é constituída por várias partições, a partição 5 está relacionado com a parte de navegação do GPS, como se ilustra na figura 87.

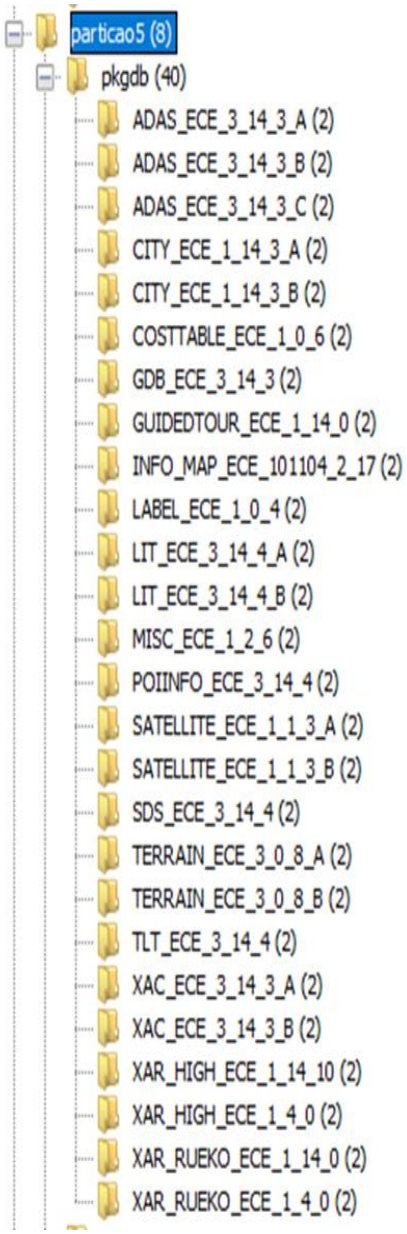


Figura 87 - Partição 5 modelo CIC

A partição 8 está relacionada com a parte de contactos telefónicos, chamadas, SMS, e-mail, tarefas, calendário, browser, cookies e dados do Bluetooth, conforme apresenta a figura 88.

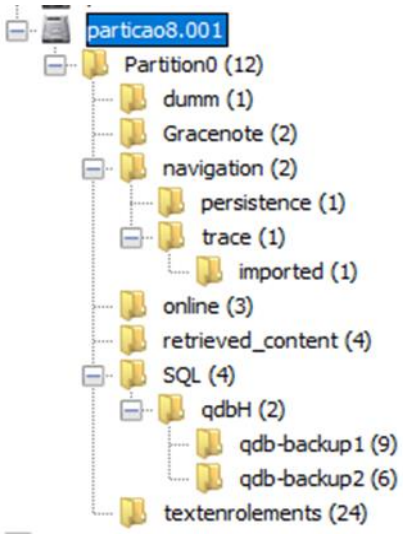


Figura 88 - Partição 8 modelo CIC

A partição 9 está relacionada com a parte de entretenimento áudio e vídeo, como se ilustra na figura 89.

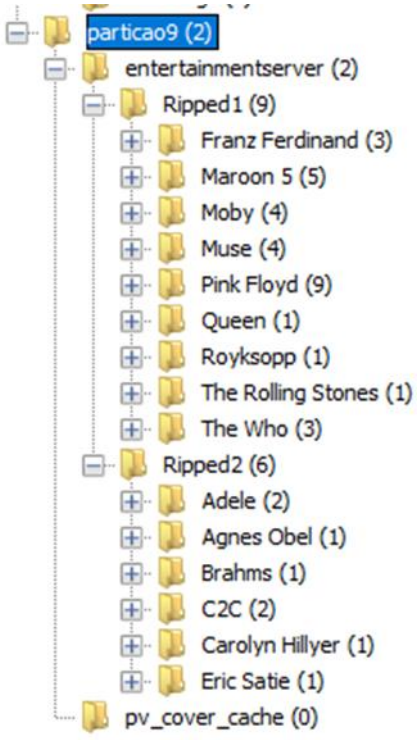


Figura 89 - Partição 9 modelo CIC

4.9.Importação dos dados para o Autopsy

Neste momento temos as partições montadas de todos os disco rígidos para serem analisadas na ferramenta Autopsy, para isso, temos de importar as partições para o Autopsy.

Para criar um caso no Autopsy, selecionamos New Case, como ilustra a figura 90.

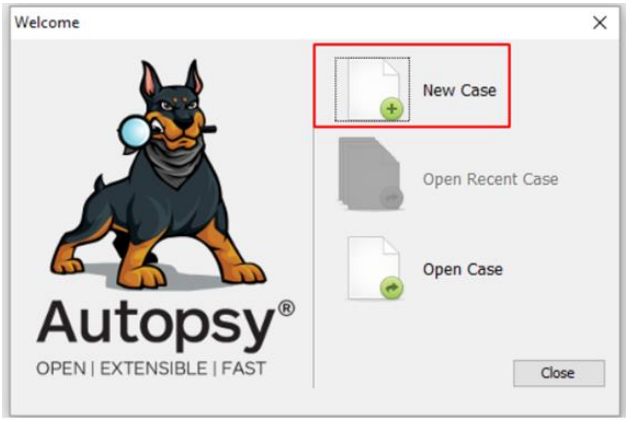


Figura 90 - Criação de um novo caso

Em seguida, preenchemos todas as informações necessárias, nome do caso e escolhemos uma pasta para guardar todos os dados num só lugar, como ilustra a figura 91.

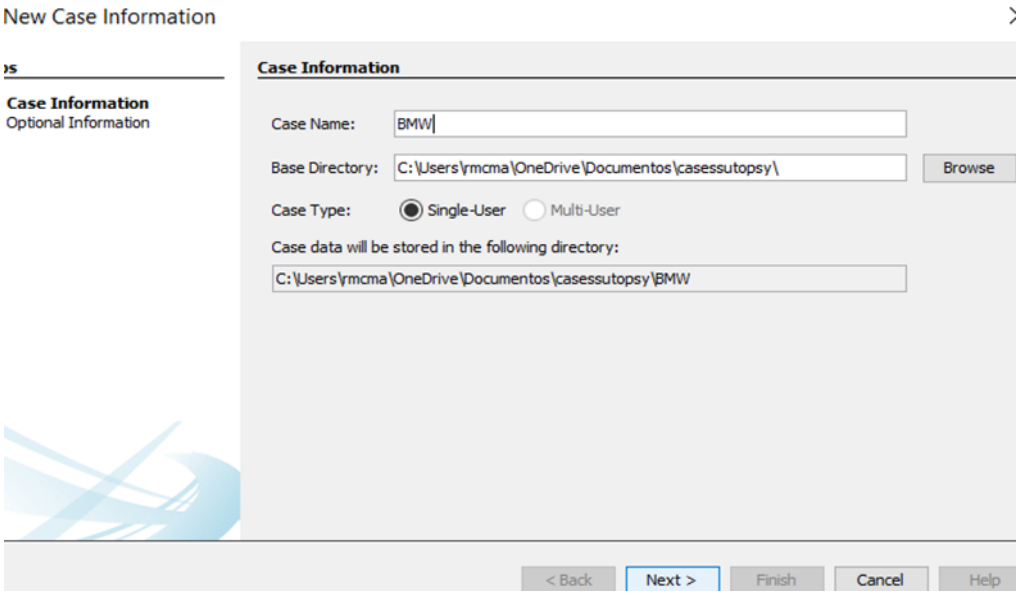


Figura 91 - Informações do novo caso

Nesta etapa descrita na figura 92, podemos adicionar outras informações opcionais sobre o caso que estamos a analisar.

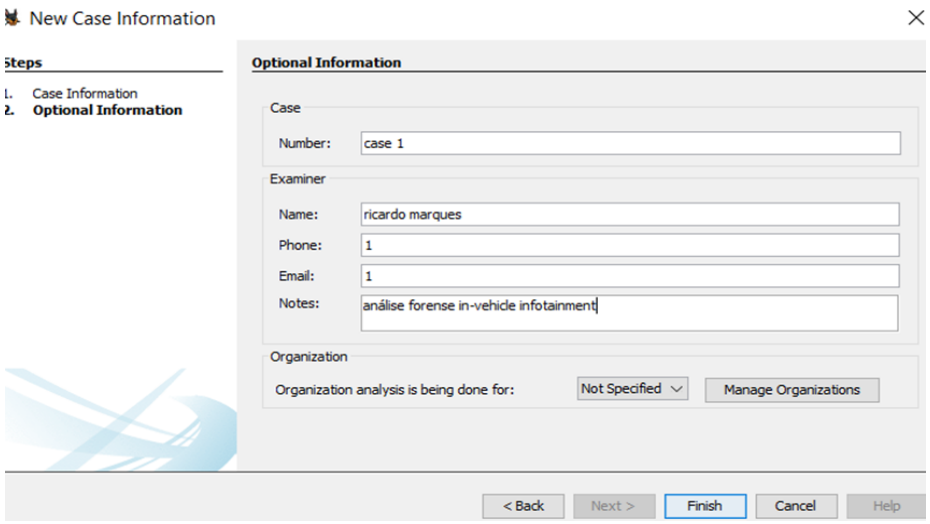


Figura 92 - Informações opcionais

Depois do caso criado, a próxima etapa é adicionar uma fonte de dados para ser o alvo da análise, no Autopsy é possível adicionar vários tipos de fontes de dados. Para importar as partições para a ferramenta Autopsy, primeiro temos de selecionar que tipo de dados vamos selecionar, neste caso escolhemos “Logical Files”, para adicionar as partições como ilustra a figura 93. O Autopsy não reconhece sistema de ficheiros QNX4 e QNX6, logo, não é possível analisar as partições QNX usando a opção “Disk Image”.

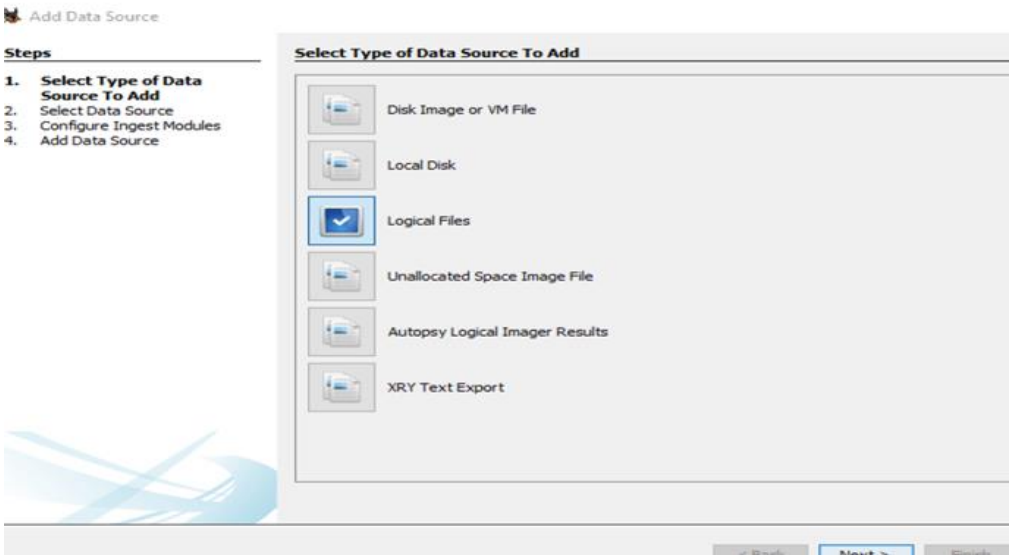


Figura 93 - Adicionar vários tipos de dados

Depois selecionamos Add para adicionar as partições. Nesta fase importamos as 4 partições do sistema IVI NBT, como podemos ver na figura 94 foram importadas as 4 partições.

Por exemplo a pasta diskp1, corresponde a uma partição, que foi montada manualmente numa diretoria no kali Linux e depois foram copiados todos os ficheiros contidos nessa diretoria para uma pasta criada no Windows com o nome diskp1, depois foi importada cada pasta que corresponde a uma partição para o Autopsy.

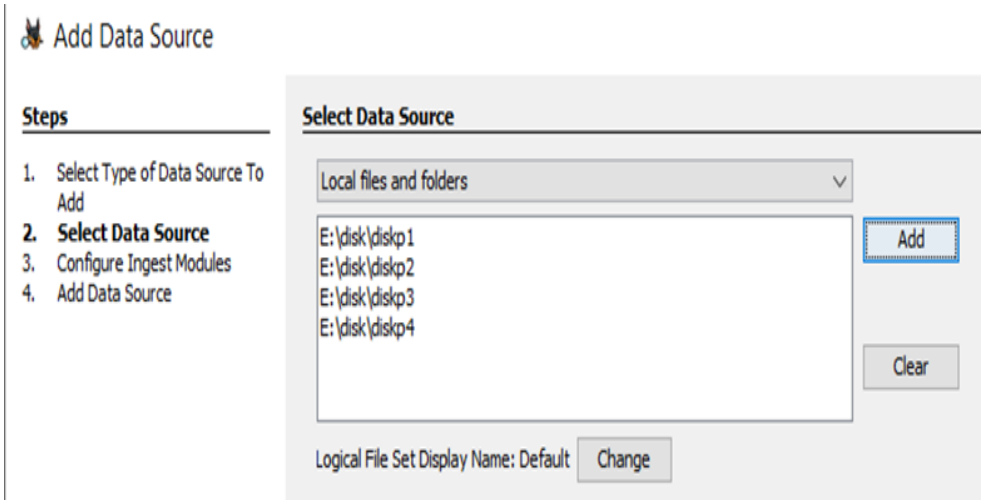


Figura 94 - Carregar as pastas das partições do sistema IVI NBT

Nesta fase vamos importar as partições do sistema IVI CIC, como podemos ver na figura 95 foram importadas 7 partições.

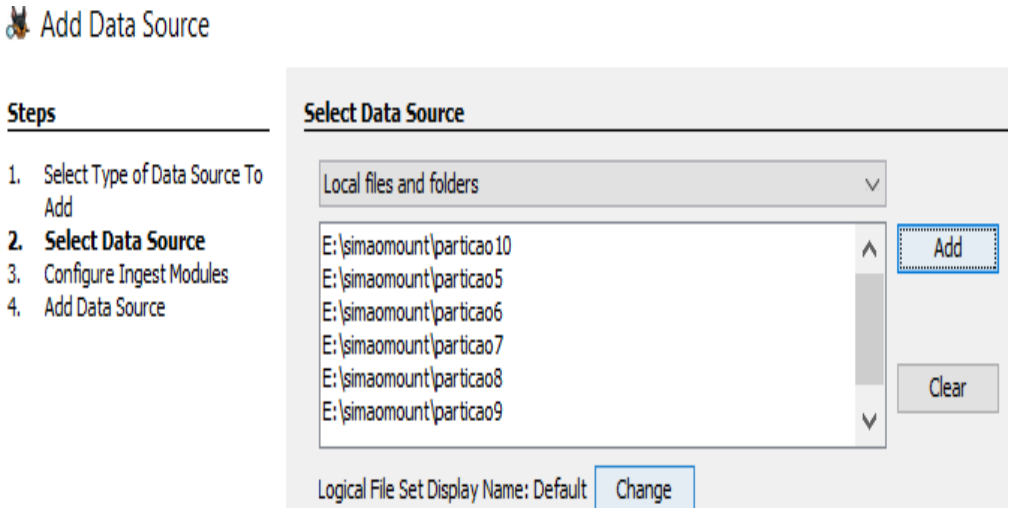


Figura 95 - Carregar as pastas das partições do sistema IVI CIC

As figuras 96 e 97 demonstram que a fonte de dados foi adicionada corretamente para proceder à análise.

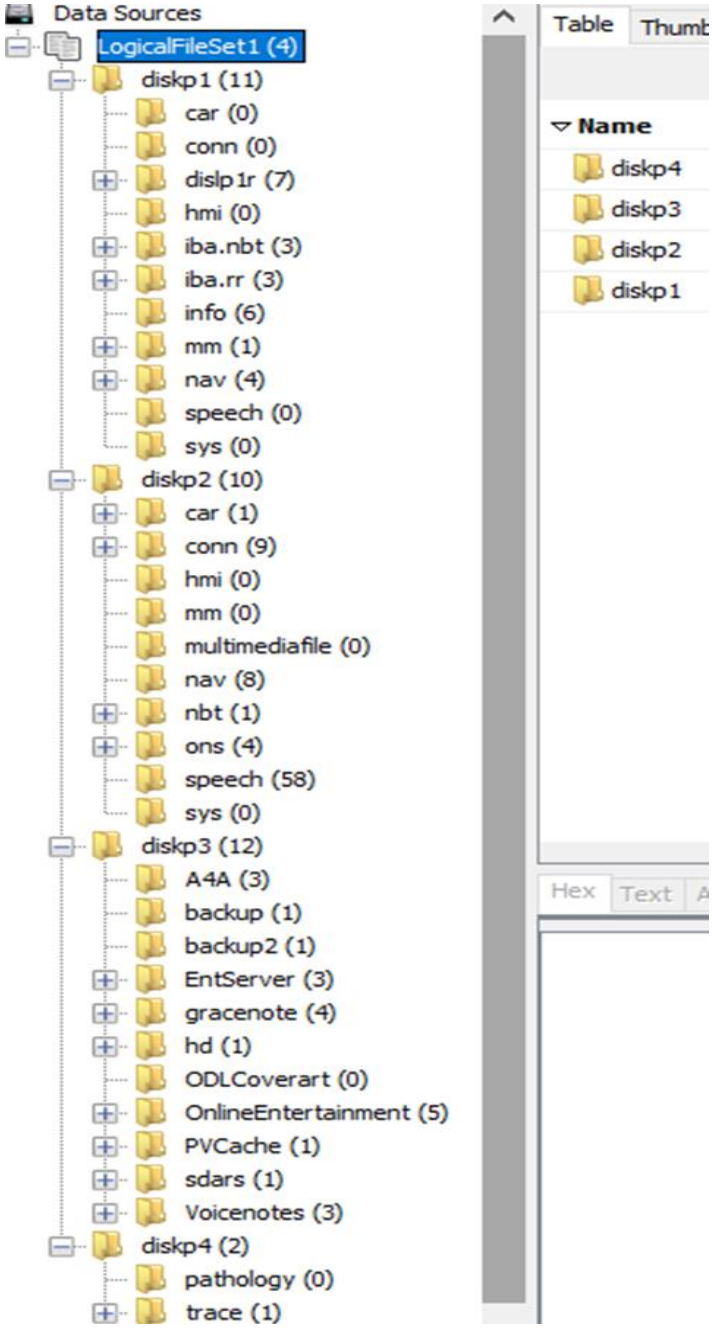


Figura 96 - Pastas do sistema IVI NBT

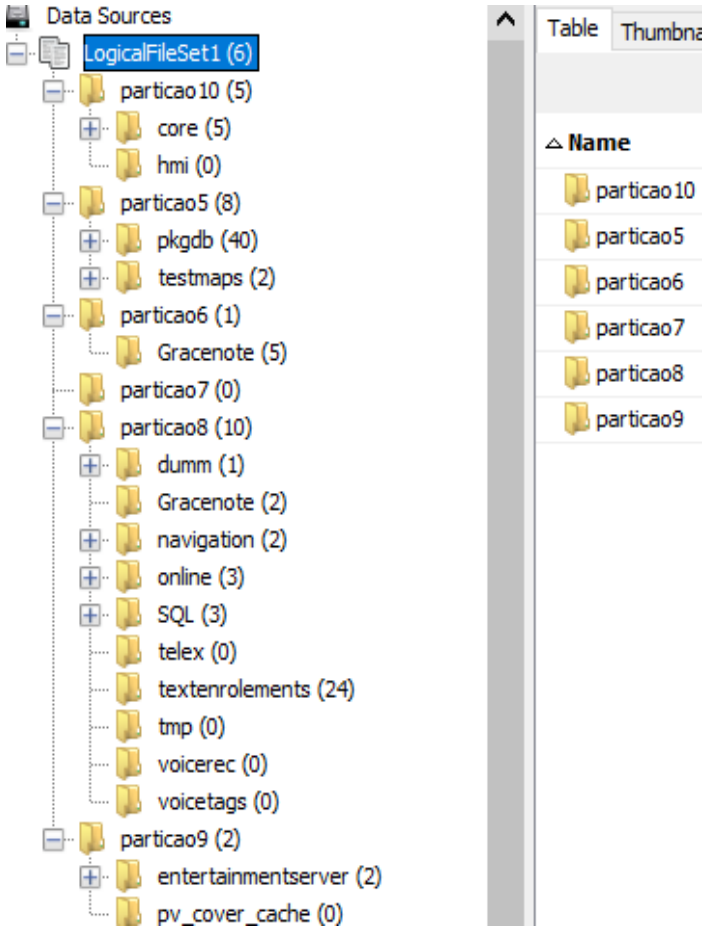


Figura 97 - Pastas do sistema IVI CIC

4.10. Base de dados SQLite modelos NBT e CIC da BMW

Os sistemas IVI da BMW usam bases de dados SQLite para guardar informação, é um formato leve para os sistemas IVI, de forma que seja fácil e rápido aceder aos dados. Uma das vantagens destes sistemas IVI usarem a base de dados SQLite, é por estas ocuparem pouco espaço. Do ponto de vista da análise forense digital estas bases de dados são muito importantes, possuem muita informação relevante.

Os dados dentro de uma base de dados SQLite são simples de explorar. Precisa-se apenas da base de dados e uma ferramenta, como o DB Browser for SQLite para consultá-la.

4.10.1. Base de dados do modelo NBT

As tabelas 4 e 5 mencionam as bases de dados mais importantes do sistema IVI NBT, da marca BMW do modelo série 5 e série 7, ambos do ano 2017, e a localização das bases de dados nas diversas partições presentes no sistema operativo QNX.

Tabela 4 -Sistema IVI NBT da marca BMW modelo serie 5 ano 2017

Sistema IVI NBT marca BMW modelo serie 5 ano 2017	
Base de dados	**Localização do ficheiro**
BrowserUrls	diskp2/conn/browser/BrowserUrls.db
cookie	diskp2/conn/browser/bon/cookie/cookie.db
contactbook_20120928	diskp2/conn/contactbook/contactbook_20120928.db
pim01	diskp2/conn/pim_bt/pim01.db
f288251944.sqlite f287629400.sqlite f287269072.sqlite f287105230.sqlite	Ficheiros excluídos/apagados
mme	diskp3/backup/mme
pm8000102 pm8000108 pm8000096	diskp2/conn/pim_bt/pm8000102.a diskp2/conn/pim_bt/ pm8000108a diskp2/conn/pim_bt/ pm8000096.a

Tabela 5 - Sistema IVI NBT da marca BMW modelo serie 7 ano 2017

Sistema IVI NBT marca BMW modelo serie 7 ano 2017	
Base de dados	**Localização do ficheiro**
BrowserUrls	p2/conn/browser/BrowserUrls.db
contactbook_20120928	p2/conn/contactbook/contactbook_20120928.db
mme	p3/backup/mme
pim01	p2/conn/pim_bt/pim01.db
pm8000002 pm8000004	p2/conn/pim_bt/pm8000004.a p2/conn/pim_bt/pm8000002.a

A seguir é explicado que tipo de dados possui cada base de dados e quais as tabelas e colunas mais importantes para a análise forense digital. Este conhecimento é muito importante para criar os dois plugins ingest modules para a ferramenta Autopsy.

Base de dados pm8000096, pm8000102, pm8000108

A única tabela nestas três bases de dados com importância a nível forense, é a tabela CALLSTACKS.

A tabela CALLSTACKS guarda o registo de chamadas do utilizador quando tem o telefone sincronizado com o sistema IVI. Cada linha corresponde a uma chamada, e é constituída

pelo número de telefone, a hora da chamada e o nome do contacto. A Tabela 6 apresenta os campos principais analisados desta tabela.

Tabela 6 – Campos da Base de dados da Tabela CALLSTACKS

Campos/colunas	Tipo	Descrição
ID	Integer	Identificador
FN	Varchar	Nome
TEL_NR	Varchar	Número de telefone
TIMESTAMP	Datetime	Data e hora

A figura 98, ilustra as colunas da tabela CALLSTACKS.

ID	FN	TEL_NR	TIMESTAMP
...	...	Filter	Filter
1	Wa	+447736...	2017-02-11 22:09:38
2	Пан	+380(50...	2017-02-11 18:47:56
3	Пан	+380(50...	2017-02-10 19:00:03
4	Ма	+380503...	2017-02-09 19:14:28
5	Пан	+380(50...	2017-02-09 19:13:50

Figura 98 - Tabela CALLSTACKS

A figura 99, ilustra a estrutura da tabela CALLSTACKS da Base de dados pm8000096, pm8000102, pm8000108.

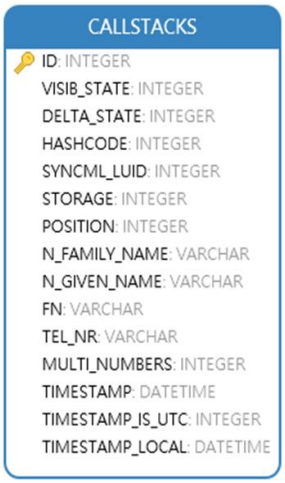


Figura 99 - estrutura da tabela CALLSTACKS

Base de dados pim01

A base de dados pim01, é usada para identificar o telefone que se ligou ao sistema IVI através do Bluetooth. A tabela CE_DEVICE_INFO, possui a informação do Bluetooth address, modelo, IMEI, IMSI do telefone que esteve sincronizado com o sistema IVI.

O Bluetooth address é um número que identifica exclusivamente um dispositivo Bluetooth de um determinado telefone.

O IMEI é constituído por 15 dígitos, os primeiros seis dígitos, denominados como Type Allocation Code (TAC), indicam o local onde o telefone foi criado. Os dois seguintes dígitos, Final Assembly Code (FAC), permitem saber quem é o fabricante. Completam a lista o número de série um dígito verificador. Um dos usos mais comuns do IMEI é para bloquear o telefone em caso de perda ou roubo. Basta contactar para a operadora e fornecer o código e uma identificação, o bloqueio será imediato. O número também pode ser útil na hora de registar uma ocorrência em caso de furto e roubo de um telefone. A figura 100 ilustra a composição do número do EMEI [49].

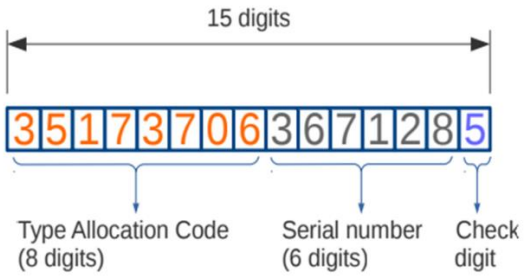


Figura 100 - Composição do n.º do EMEI [50]

O cartão SIM contém um conjunto de códigos nomeadamente o IMSI. Este código é o número único padronizado internacionalmente para identificar um assinante móvel, é constituído por 15 dígitos e identifica o utilizador na rede móvel a que está ligado. Este código está registado digitalmente no cartão [50].

Por exemplo, o número IMSI 310410123456789, o número 310 corresponde ao código do país móvel (MCC), 410 código de rede móvel (MNC), e o número 123456789 corresponde ao número de identificação de assinatura móvel (MSIN), a figura 101, ilustra a composição do número IMSI [51].

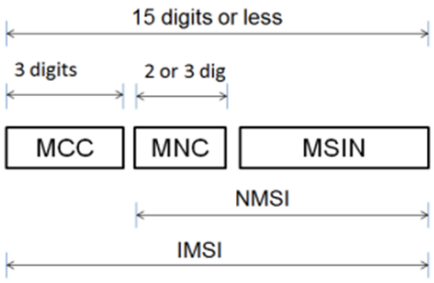


Figura 101 - Composição do número IMSI [50]

A Tabela 7 mostra os campos principais analisados da tabela CE_DEVICE_INFO.

Tabela 7 - Colunas da tabela CE_DEVICE_INFO

Campos/coluna	Tipo	Descrição
PID	Integer	Identificador
INFO KEY	Varchar	Tipo de informação (Bluetoothaddress, Model, Vendor, EMEI e IMSI)
INFO_VALUE	Varchar	Número de identificação (Bluetoothaddress, Model, Vendor, EMEI e IMSI)

A figura 102, ilustra os campos da tabela CE_DEVICE_INFO.

PID	INFO_KEY	INFO_VALUE
Filter	Filter	Filter
513	BluetoothAdd...	70:70:0D:8F:...
514	Model	PHONE_MOD...
515	Vendor	PHONE_VEN...
516	IMEI	35384508055...
517	IMSI	23410037337...

Figura 102 - Tabela CE_DEVICE_INFO

A figura 103, ilustra a estrutura da tabela CE_DEVICE_INFO da base de dados pim01.

Figura 103 - estrutura da tabela CE_DEVICE_INFO

Base de dados f287105230, f287269072, f287629400, f288251944

Esta base de dados possui tabelas que contêm dados relativos a mensagens baseadas no telefone sincronizado com o sistema IVI. A tabela messages guarda os dados de mensagens SMS, como o texto da mensagem que é guardado no campo do subject, contém também o número de telefone que enviou a mensagem e a data do envio. O campo date regista a data e hora em que a mensagem SMS foi enviada, o campo fromPhoneNumber menciona o número de quem enviou a mensagem SMS.

A Tabela 8 demonstra os campos principais da tabela messages.

Tabela 8 - Campos da tabela messages

Campos/coluna	Tipo	Descrição
id	Integer	Identificador
fromPhoneNumber	Varchar	Número de telefone
date	Varchar	Data
subject	Varchar	Texto da mensagemm

A figura 104, ilustra a tabela messages.

id	fromPhoneNumber	date	subject
Fi...	Filter	Filter	Filter
237	+447 7551	20170624082607	Its not fo...
238	+447 7551	20170624082434	U want ...
239	+447 7551	20170624081807	U free at...
240	+447 7551	20170624072048	Wher
241	+447 8051	20170624050945	Have you...
242	+447 6846	20170624044458	Alright b...
243	+447 6846	20170624044414	Bro just ...

Figura 104 - Tabela messages

A Figura 105 ilustra a estrutura da tabela messages da Base de dados f287105230, f287269072, f287629400, f288251944

Figura 105 - Figura estrutura da tabela messages

Base de dados contactbook_20120928

Esta base de dados tem várias tabelas importantes a nível forense, nomeadamente as tabelas **contact_card_phone, msg_data_phone**, **address_phone**, **phone_data_phone,** estas tabelas estão relacionadas com os contatos. Esta base de dados também possui uma tabela relativa ao MAC address do bluetooth.

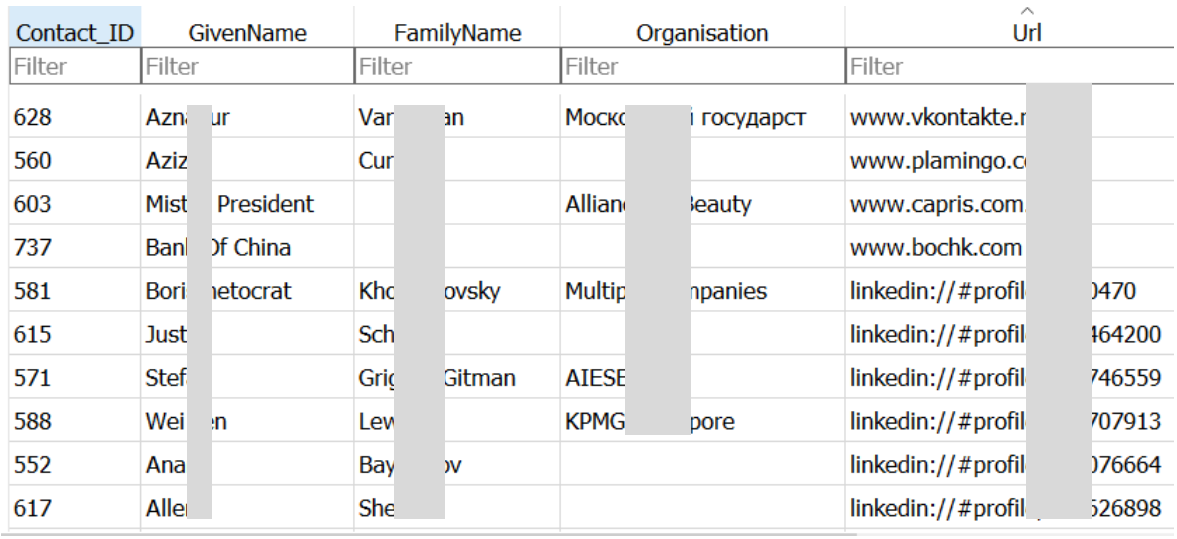
Cada linha na tabela contact_card_phone corresponde a uma entrada na lista de contatos do telefone do utilizador. Esta tabela contêm os dados básicos de identificação do contacto. A tabela address_phone, está relacionada com os dados da residência do contato, menciona a rua, cidade país e código postal referente a cada número de telefone. A tabela msg_data_phone possui o email referente a cada contacto. A tabela phone_data_phone contém os números de telefone de cada contacto.

A tabela 9 possui os campos principais da tabela contact_card_phone da base de dados contactbook_20120928.

Tabela 9 - Campos da tabela contact_card_phone

Campos/coluna	Tipo	Descrição
Contact_ID	Integer	Identificador
GivenName	Varchar	Nome
FamilyName	Varchar	Apelido
Url	Varchar	URL
Organisation	Varchar	Organização

A figura 106, ilustra a tabela contact_card_phone.



Contact_ID	GivenName	FamilyName	Organisation	Url
Filter	Filter	Filter	Filter	Filter
628	Azn ur	Var an	Моск государст	www.vkontakte.r
560	Aziz	Cur		www.plamingo.c
603	Mist President		Allian Beauty	www.capris.com
737	Banl Of China			www.bochk.com
581	Bori netocrat	Kho ovsky	Multip npanies	linkedin://#profil 0470
615	Just	Sch		linkedin://#profil 464200
571	Stef	Grig Gitman	AIESE	linkedin://#profil 746559
588	Wei n	Lew	KPMG pore	linkedin://#profil 707913
552	Ana	Bay ov		linkedin://#profil 076664
617	Alle	She		linkedin://#profil 526898

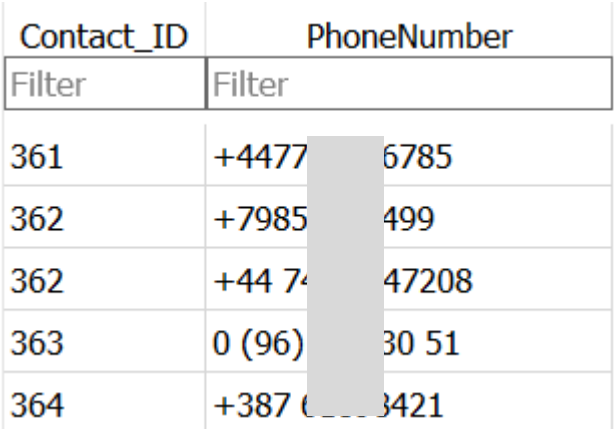
Figura 106 - Tabela contact_card_phone

A tabela 10 possui os campos principais da tabela phone_data_phone da base de dados contactbook_20120928.

Tabela 10 - Campos da tabela phone_data_phone

Campos/coluna	Tipo	Descrição
Contact_ID	Integer	Identificador
PhoneNumber	Varchar	Número de telefone

A figura 107, ilustra a tabela phone_data_phone.



Contact_ID	PhoneNumber
Filter	Filter
361	+4477 6785
362	+7985 499
362	+44 74 47208
363	0 (96) 30 51
364	+387 (3421

Figura 107 - Tabela phone_data_phone.

A tabela 11 possui os campos principais da tabela address_phone da base de dados contactbook_20120928.

Tabela 11 - Campos da tabela address_phone

Campos/coluna	Tipo	Descrição
Contact_ID	Integer	Identificador
StreetHousenumber	Varchar	Rua
City	Varchar	Cidade
Country	Varchar	País
Postalcode	Varchar	Código postal

A figura 108, ilustra a tabela address_phone.

Contact_ID	City	Country	Postalcode	StreetHousenumber
Filter	Filter	Filter	Filter	Filter
423	London	Велика Брит...	SE2 RG	21 Pit rescent
500	Одесса	Украина		Маяк о 1
530	Odessa	Ukraine		арме 17 64
547	Cupertino	United States	950	1 Infi op
548	Les Escaldes	Andorra	AD	

Figura 108 - Tabela address_phone

A tabela 12 possui os campos principais da tabela msg_data_phone da base de dados contactbook_20120928.

Tabela 12 - Campos da tabela msg_data_phone

Campos/coluna	Tipo	Descrição
Contact_ID	Integer	Identificador
EmailAddr	Varchar	E-mail

A figura 109, ilustra a tabela msg_data_phone.

Contact_ID	EmailAddr
Filter	Filter
362	abbasmamed...
382	laym0n@ram...
424	e-esaulov@uk...
427	zulfiya.nazyro...
482	aliyance.7@g...
500	7436538@ma...

Figura 109 - Tabela msg_data_phone

A tabela bluetooth possui os dados do MAC address do Bluetooth correspondente a cada telefone sincronizado com o sistema IVI, conforme consta na tabela 13.

Tabela 13 - Campos da tabela Bluetooth

Campos/coluna	Tipo	Descrição
Origin	Integer	Identificador
BtAddress	Varchar	MAC address Bluetooth

A figura 110, ilustra a tabela bluetooth.

Origin	BtAddress
Filter	Filter
7	70:70:0D:8F:07:DC
5	2C:33:61:45:D0:49
4	DC:2B:2A:56:9C:BF

Figura 110 - Tabela bluetooth

A figura 111 ilustra a estrutura da tabela Estrutura da base de dados contactbook_20120928.

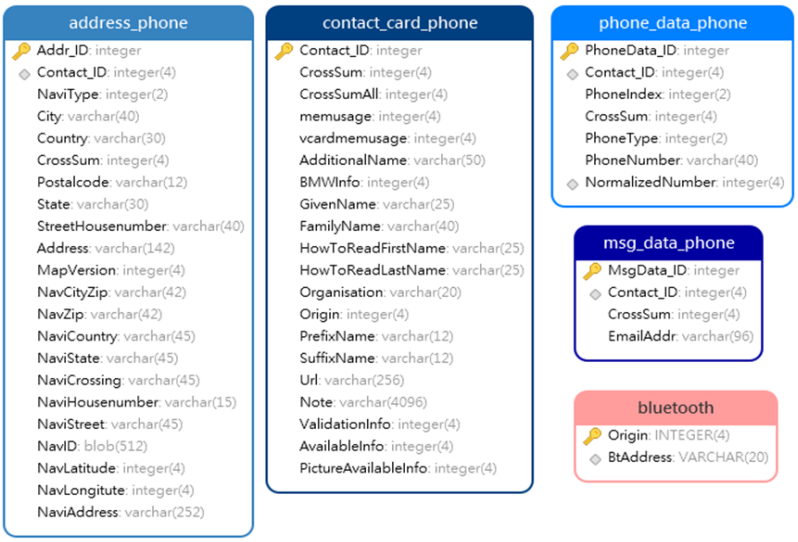


Figura 111 - Estrutura da base de dados contactbook_20120928

Base de dados BrowserUrls

Esta base de dados possui informação relativa ao histórico de navegação na web. A tabela urls e visits contém o histórico da web, que websites foram pesquisados. A tabela url, no campo title e url, apresenta o website visitado e a tabela visits no campo datevisits, contém a data desse mesmo acesso ao website.

A tabela 14 apresenta os principais campos da tabela urls da base de dados BrowserUrls.

Tabela 14 - Campos da tabela urls

Campos/coluna	Tipo	Descrição
Id	Integer	Identificador
Title	Varchar	Nome do site
url	Varchar	Url do site

A figura 112, ilustra a tabela urls.

id	url	title
F...	Filter	Filter
1	http://www.google.com/	http://www.google.com/

Figura 112 - Tabela urls

A tabela 15 apresenta os principais campos da tabela visits da base de dados BrowserUrls

Tabela 15 - Campos da tabela visits

Campos/coluna	Tipo	Descrição
datevisits	Datetime	Data e hora da consulta

A figura 113, ilustra a tabela visits, neste caso estava sem dados.

Figura 113 - Tabela visits

A figura 114 apresenta a estrutura da base de dados BrowserUrls.

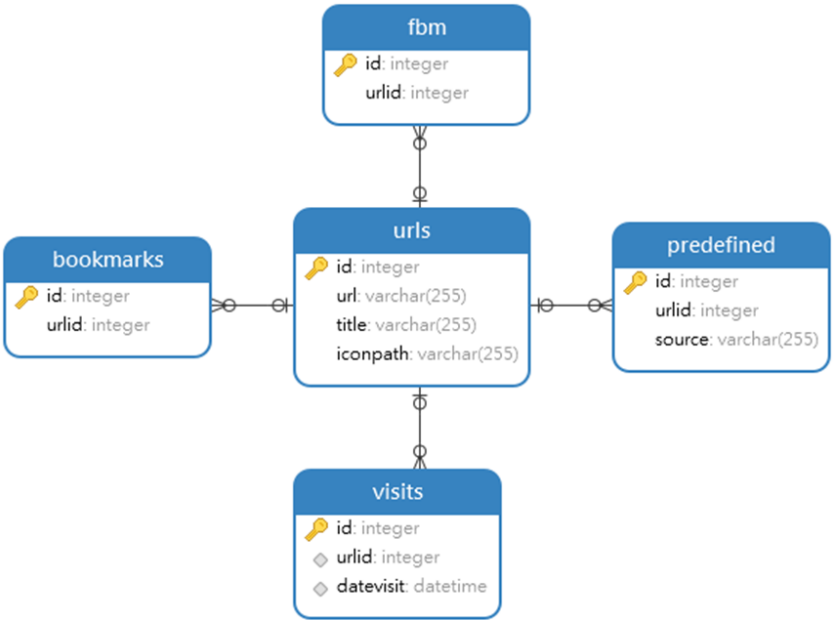


Figura 114 - Estrutura da base de dados BrowserUrls

Base de dados cookie

A tabela 16 apresenta os campos da tabela cookies, que contém o registo de hosts no campo host, o caminho no campo path e o último acesso no campo lastaccessed.

Tabela 16 - Campos da tabela cookies

Campos/coluna	Tipo	Descrição
name	Text	Nome
host	Text	Host
path	Text	Caminho
lastAccessed	Double	Data e hora do último acesso

A figura 115, ilustra a tabela cookies.

name	host	path	lastAccessed
Filter	Filter	Filter	Filter
VEHICLEAUTH	b2v.bmwgroup.de	/DUMMY_PATH	1489327221.33596
JSESSIONID	b2v.bmwgroup.de	/DUMMY_PATH	1489327221.63296
VEHICLEAUTH	cngw.bmwgroup.de	/DUMMY_PATH	1489327221.63496
JSESSIONID	cngw.bmwgroup.de	/DUMMY_PATH	1489327221.63896

Figura 115 - Tabela cookies

A figura 116 apresenta a estrutura da tabela cookies da base de dados cookies.

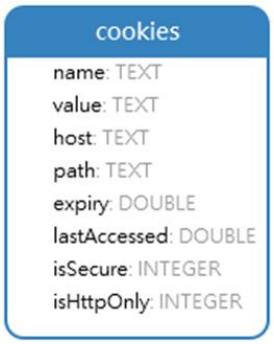


Figura 116 - Estrutura da tabela cookie

Base de dados mme

A tabela mediastores mantém o registo da sincronização de vários dispositivos, desde telefones, CD/DVD, iPods e pendrives sincronizados com o sistema IVI. Cada linha corresponde a um determinado dispositivo ligado, no campo lastseen refere data e hora da última ligação, os campos msname e name referem o nome do dispositivo sincronizado. O campo identifier menciona a identificação atribuída a cada dispositivo sincronizado. Esta tabela tem mais campos, mas sem interesse forense e alguns campos estão vazios.

A tabela 17 apresenta os principais campos da tabela mediastores da base de dados mme.

Tabela 17 - Campos da tabela mediastores

Campos/coluna	Tipo	Descrição
Msid	Integer	Identificador
Lastseen	Integer	Data e hora do último acesso
Msname	Text	Tipo de dispositivo
Name	Text	Nome do dispositivo
Identifier	Text	Identificação do dispositivo
Mountpath	Text	Caminho

A figura 117, ilustra a tabela mediastores

msid	lastseen	mssname	name	identifier	mountpath
Fil...	Filter	Filter	Filter	Filter	Filter
1	1498990899643...	internet	SocketDMBSl...	/dev/socket	/dev/socket
6	1389956516853...	mediafs2wire	Charlotte's iP...	88:53:95:B8:19:33-1	
7	1389956516018...	ipod	Charlotte's iP...	DQGJK12UDTC0	
8	1401984309235...	mediafs2wire	C2105	B4:52:7D:D4:19:28	
9	1394564684627...	mediafs2wire	Esmee's iPhone	90:B9:31:81:C9:25-2	
10	1406033894603...	ipod	Esmee's iPhone	DNPLMDLDFFGC	
11	1406033895367...	mediafs2wire	Esmee's iPhone	90:B9:31:81:C9:25-1	
12	1395164374528...	mediafs2wire	Esmee's iPhone	90:B9:31:81:C9:25	
13	1403433679529...	audiocd	CD/DVD	ce12501f	
14	1419939107987...	audiocd	WHAT HAVE ...	8d0aa50c	

Figura 117 - Tabela mediastores

A tabela categorydata_custom possui o registo de músicas e artistas musicais na coluna name. São dados que têm alguma relevância pois permitem tirar conclusões sobre o perfil de um indivíduo.

A tabela 18 apresenta os principais campos da tabela categorydata_custom da base de dados mme.

Tabela 18 - Campos da tabela categorydata_custom

Campos/coluna	Tipo	Descrição
Name	Text	Nome dos artistas musicais

A figura 118, ilustra a tabela categorydata_custom.

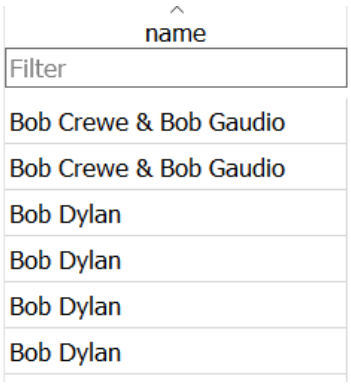


Figura 118 - Tabela categorydata_custom

A tabela software_info da base de dados mme, possui a versão do software, como apresenta na tabela 19.

Tabela 19 - Campos da tabela software_info

Campos/coluna	Tipo	Descrição
Version	Text	Versão do software

A figura 119, ilustra a tabela software_info

Figura 119 - Tabela software_info

A tabela usbdevicedetails da base de dados mme, demonstra os detalhes da porta usb que esteve ligado ao sistema IVI e a data/hora da última sincronização, como apresenta na tabela 20.

Tabela 20 - Campos da Tabela usbdevicedetails

Campos/coluna	Tipo	Descrição
deviceserialno	Text	Número de serie do USB
lastseen	Integer	Data e hora do último acesso

A figura 120, ilustra a tabela usbdevicedetails.

deviceserialno	lastseen
Filter	Filter
F2LS93GWHFYH	0
07AA140886E835AD	0

Figura 120 - Tabela usbdevicedetails

A tabela folders da base de dados mme, possui os nomes das pastas bem como o seu caminho e última sincronização, como demonstra na tabela 21.

Tabela 21 - Campos da tabela folders

Campos/coluna	Tipo	Descrição
foldername	Text	Nome da pasta
last_sync	Integer	Dara e hora do último acesso
basepath	Text	Caminho da pasta

A figura 121, ilustra a tabela folders.

last_sync	foldername	basepath
Filter	Filter	Filter
14971441316...	urban	/a s erotic/urban/
14901170038...	tmp	/tmp/
14971441316...	temp cads for china	/temp cads for china/
14971441316...	ss11 images baby glam	/ss11 images baby glam/
14971441316...	sally's mold cups	/temp cads for china/Gel boost cup photos/sally's mold cups/
14971441316...	photos from China	/photos from China/
14971441316...	photos	/photos/

Figura 121 - Tabela folders

A tabela library_albums da base de dados mme, possui dados dos álbuns que o utilizador deste sistema IVI possui, como demostra na tabela 22.

Tabela 22 - Campos da tabela library_albums

Campos/coluna	Tipo	Descrição
Álbum	Text	Nome do álbum de música

A figura 122, ilustra a tabela library_albums.

Figura 122 - Tabela library_albums

A tabela library_artists, da base de dados mme, possui dados de artistas musicais que o utilizador deste sistema IVI possui, como demostra na tabela 23

Tabela 23 - Campos da tabela library_artists

Campos/coluna	Tipo	Descrição
artist	Text	Nome de artistas musicais

A figura 123, ilustra a tabela library_artists.

Figura 123 - tabela library_artists

A figura 124 identifica a estrutura da base de dados mme, com todas as tabelas mencionas anteriormente.

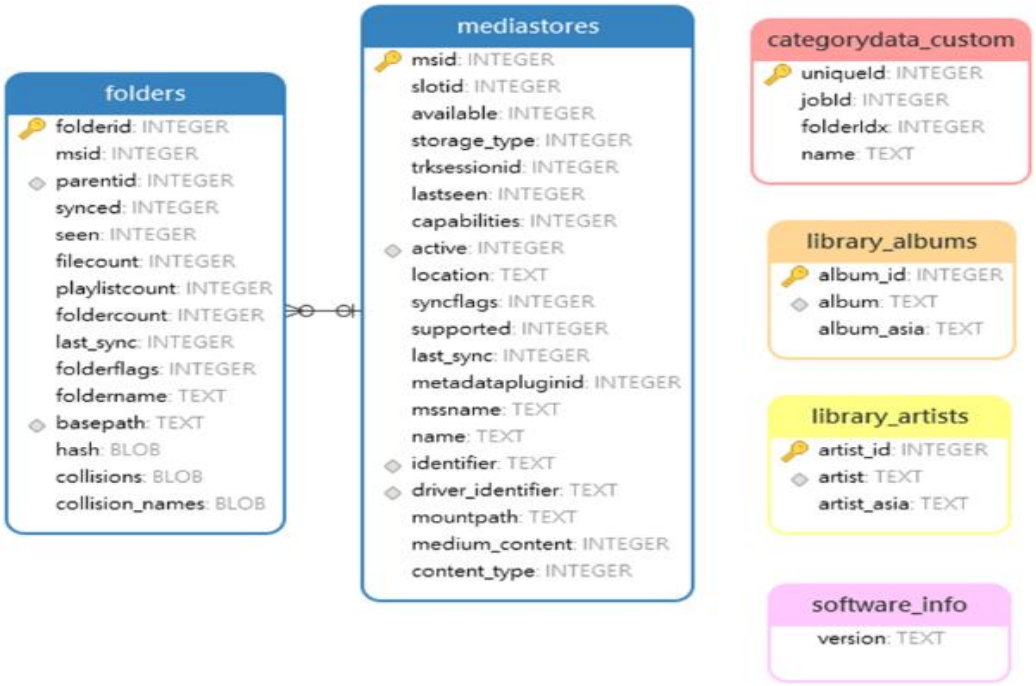


Figura 124 - Estrutura da base de dados mme

4.10.2. Base de dados do modelo CIC

As tabelas 24 e 25, identificam as bases de dados mais importantes do sistema IVI CIC da marca BMW, do modelo série 3 do ano 2012 e 2010, e a localização das bases de dados nas diversas partições.

Tabela 24 - Sistema IVI CIC da marca BMW modelo serie 3 ano 2012

Sistema IVI CIC marca BMW modelo serie 3 ano 2012	
Base de dados	Localização do ficheiro
contactbook_20101214	particao8/SQL/contactbook_20101214.db
mme	particao8/SQL/qdbH/qdb-backup2/mme
mme_library	particao8/SQL/qdbH/qdb-backup1/mme_library
mme_custom	particao8/SQL/qdbH/qdb-backup1/mme_custom
mme_library	particao8/SQL/qdbH/qdb-backup1/mme_library
mme_library	particao8/SQL/qdbH/qdb-backup1/mme_library

Tabela 25 - Sistema IVI CIC da marca BMW modelo serie 3 ano 2012

Sistema IVI CIC marca BMW modelo serie 3 ano 2010	
Base de dados	**Localização do ficheiro**
contactbook_20101214	particao8/SQL/contactbook_20101214.db
mme	particao8/SQL/qdbH/qdb-backup2/mme
mme_library	particao8/SQL/qdbH/qdb-backup1/mme_library
mme_custom	particao8/SQL/qdbH/qdb-backup1/mme_custom
mme_library	particao8/SQL/qdbH/qdb-backup1/mme_library
mme_library	particao8/SQL/qdbH/qdb-backup1/mme_library

A seguir é explicado, que tipo de dados tem cada base de dados e quais as tabelas e colunas mais importantes para a análise forense digital.

Base de dados contactbook_20101214

Esta base de dados possui várias tabelas importantes a nível forense, nomeadamente a tabela **contact_card_phone, msg_data_phone, address_phone, phone_data_phone,** estas tabelas estão relacionadas com os contatos, também foi analisada a tabela Bluetooth, que possui o MAC address.

Cada linha na tabela contact_card_phone corresponde a uma entrada na lista de contatos do telefone do utilizador. Esta tabela contêm os dados básicos de identificação do contacto. A tabela address_phone, está relacionada com os dados da residência do contato, menciona a rua, cidade país e código postal referente a cada número de telefone. A tabela msg_data_phone possui o email referente a cada contacto. A tabela phone_data_phone contém os números de telefone de cada contacto.

A tabela 26 possui os campos principais da tabela contact_card_phone da base de dados contactbook_20101214.

Tabela 26 - Campos da tabela contact_card_phone

Campos/coluna	Tipo
Contact_ID	Integer
GivenName	Varchar
FamilyName	Varchar
Url	Varchar
Organisation	Varchar

A tabela 27 possui os campos principais da tabela phone_data_phone da base de dados contactbook_20101214.

Tabela 27 - Campos da tabela phone_data_phone

Campos/coluna	Tipo
Contact_ID	Integer
PhoneNumber	Varchar

A tabela 28 possui os campos principais da tabela address_phone da base de dados contactbook_20101214.

Tabela 28 - Campos da tabela address_phone

Campos/coluna	Tipo
Contact_ID	Integer
StreetHousenumber	Varchar
City	Varchar
Country	Varchar
Postalcode	Varchar

A tabela 29 possui os campos principais da tabela msg_data_phone da base de dados contactbook_20101214.

Tabela 29 - Campos da tabela msg_data_phone

Campos/coluna	Tipo
Contact_ID	Integer
EmailAddr	Varchar

A Tabela bluetooth possui os dados do mac address do Bluetooth correspondente a cada telefone sincronizado com o sistema IVI, conforme consta na tabela 30.

Tabela 30 - Campos da tabela bluetooth

Campos/coluna	Tipo
Origin	Integer
BtAddress	Varchar

A figura 125 ilustra a estrutura das tabelas da base de dados contactbook_20101214.

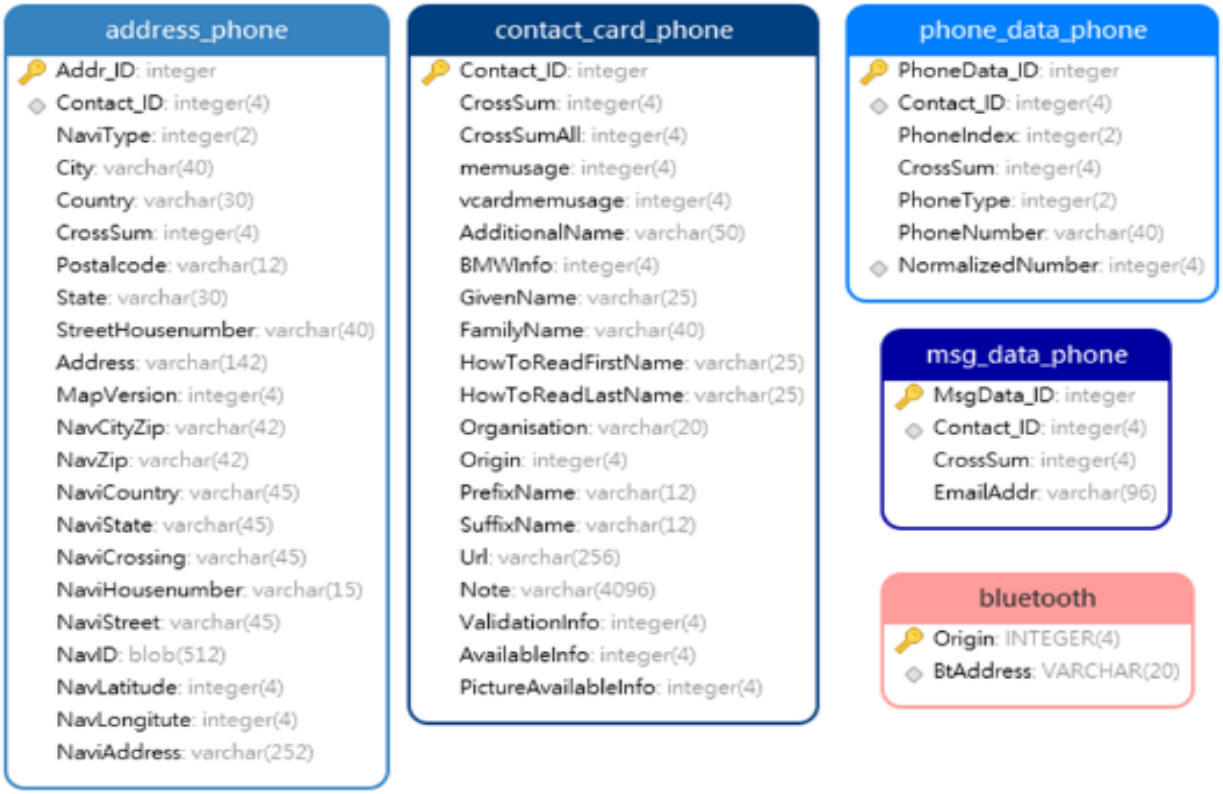


Figura 125 - Estrutura da base de dados contactbook_20101214

Base de dados mme

A tabela mediastores mantém o registo da sincronização de vários dispositivos com o sistema IVI, tais como smartphones, CD/DVD, iPods e pendrives. No campo lastseen refere data e hora da última ligação, o campo msname e name referem o nome do dispositivo sincronizado. O campo identifier menciona a identificação atribuída a cada dispositivo sincronizado. Esta tabela tem mais campos, mas sem interesse forense e alguns campos estão vazios.

Tabela 31 apresenta os principais campos da tabela mediastores da base de dados mme.

Tabela 31 - Campos da tabela mediastores

Campos/coluna	Tipo
msid	Integer
lastseen	Integer
msname	Text
name	Text
identifier	Text
mountpath	Text

Figura 126, apresenta a estrutura da tabela mediastores da base de dados mme.

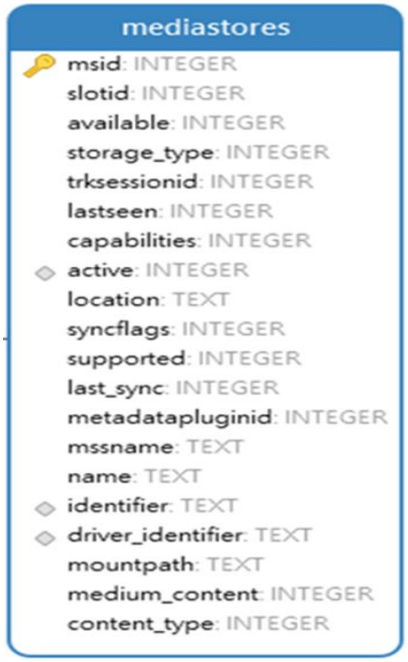


Figura 126 - Estrutura da tabela mediastores

Base de dados mme_custom

A tabela 32 apresenta os principais campos da tabela software_info da base de dados mme_custom, a qual possui a versão do software.

Tabela 32 - Campos da tabela software_info

Campos/coluna	Tipo
version	Text

A figura 127, apresenta a estrutura da tabela software_info da base de dados mme_custom.

Figura 127 - Estrutura da tabela software_info

Base de dados mme_library

A tabela 33 apresenta os principais campos da tabela library_artists da base de dados mme_library. Esta tabela library_artists possui dados de artistas musicais.

Tabela 33 - Campos da tabela library_artists

Campos/coluna	Tipo
artist	Text

A figura 128, apresenta a estrutura da tabela library_artists da base de dados mme_library.

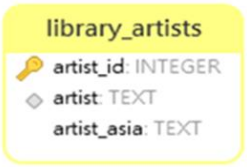


Figura 128 - Estrutura da tabela library_artists

A tabela 34 apresenta os principais campos da tabela library_albums da base de dados mme_library.

A tabela library_albums possui dados dos álbuns que o utilizador deste sistema IVI.

Tabela 34 - Campos da tabela library_albums

Campos/coluna	Tipo
album	Text

A figura 129, apresenta a estrutura da tabela library_albums da base de dados mme_library

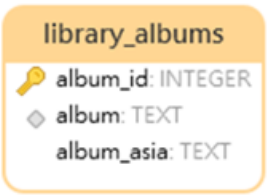


Figura 129 - Estrutura da tabela library_albums

A tabela 35 apresenta os principais campos da tabela folders da base de dados mme_library. A tabela folders possui os nomes das pastas bem como o seu caminho e última sincronização.

Tabela 35 - Campos da tabela folders

Campos/coluna	Tipo
foldername	Text
last_sync	Integer
basepath	Text

A figura 130, apresenta a estrutura da tabela folders da base de dados mme_library.

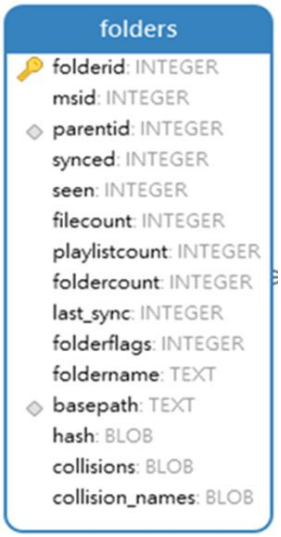


Figura 130 - Estrutura da tabela folders

5. Plugins Autopsy para análise forense dos sistemas IVI CIC e NBT

Este capítulo descreve o processo de desenvolvimento do ingest module e a sua arquitetura, sendo um dos objetivos deste projeto. A ferramenta Autopsy permite a integração de módulos que, por sua vez, adicionam outras funcionalidades à ferramenta.

Por padrão, o Autopsy vem com um conjunto inicial de módulos. Na Figura 131, é possível visualizar alguns dos módulos disponíveis que já estão inseridos no Autopsy.

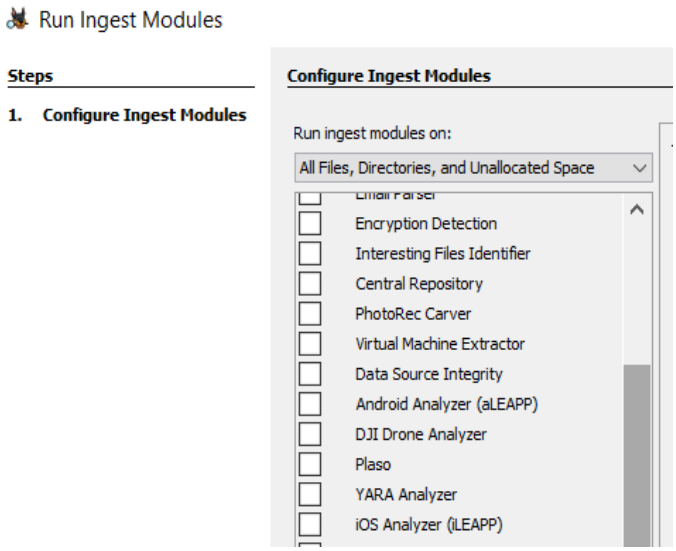


Figura 131 -Selecionar ingest module

5.1. Data source ingest module

Uma das propriedades mais importantes do Autopsy é a possibilidade de adicionar os próprios ingest module personalizados em Java ou Python (Jython).

Para desenvolver um ingest module para a ferramenta Autopsy, foi necessário um estudo de conceitos próprios da programação de módulos para Autopsy, tais como, blackboard artifact, blackboardatribute, linguagem SQL e Python. Alguns aspetos e características acerca do módulo e do seu funcionamento foram definidos e estruturados antes de se dar início à fase de desenvolvimento do mesmo.

O desenvolvimento do ingest module baseia-se no template[5] disponibilizado no GitHub. A linguagem de programação escolhida para este projeto foi o Python. Python é uma linguagem de programação orientada a objetos. O Python usa uma sintaxe clara e concisa que facilita a leitura e escrita de código, e tem grande variedade de bibliotecas disponíveis, o que torna o desenvolvimento rápido e fácil. A interpretação de código, permite a execução rápida e simplificada de scripts e programas.

O Autopsy usa Jython, para habilitar scripts em Python. Jython é uma implementação do Python que é executado na máquina virtual Java (JVM). Isto significa que, embora a sintaxe e a estrutura do código Jython sejam as mesmas do Python padrão, o Jython pode acessar a recursos da JVM, como bibliotecas Java e o ambiente de execução. No Jython os scripts são convertidos em bytecode Java e executados na JVM. Esta configuração de desenvolvimento tem, no entanto, algumas limitações: é limitada ao Python versão 2.7.

É uma das funcionalidades mais poderosas da ferramenta Autopsy, ao adicionar ingest module os utilizadores têm a possibilidade de aumentar o número de funcionalidades graças ao uso de módulos criados.

Cada modulo Python na ferramenta Autopsy deve ter a sua própria pasta, isso reduz as colisões de nomes entre outros módulos. Tool - Python Plugins é a forma para colocar o modulo no Autopsy.

Destro da pasta Python_modules foram criadas duas pastas “BMW IVI CIC Ingest Module” e “BMW IVI NBT Ingest Module”, dentro destas pastas encontram-se os ingest modules, CicIvibmwDataSourceIngestModule.py[6] e IvibmwDataSourceIngestModule.py[7].

Estes módulos ao serem executados vão permitir encontrar de forma automatizada as bases de dados SQLite mais importantes dos sistemas IVI da BMW dos modelos CIC e NBT, permitindo encontrar muitos dados relevantes para uma análise forense digital.

Antes de se dar início à fase de desenvolvimento dos ingest modules, foi necessário estudar as bases de dados que consta na secção 4.10 do capítulo 4, para definir quais as tabelas e colunas mais importantes para depois criar o ingest module.

[5] https://github.com/sleuthkit/autopsy/blob/develop/pythonExamples/dataSourceIngestModule.py/
[6] https://github.com/Ricardo2192643/BMW-IVI-CIC-Ingest-Module
[7] https://github.com/Ricardo2192643/BMW-IVI-NBT-ingest-module

5.1.1. Desenvolvimento do ingest module

A principal função destes módulos é encontrar as bases de dados SQLite e depois efetuar a análise de forma automatizada. O uso destas bibliotecas é fornecer funcionalidades adicionais para o desenvolvimento de ingest module no Autopsy. Cada biblioteca tem a sua própria função, oferecem uma ampla gama de recursos para manipular os diferentes casos, ficheiros, artefactos e outros aspetos no ambiente Autopsy. As bibliotecas que vão ser usadas no ingest module servem para permitir certas funcionalidades necessárias ao funcionamento do módulo, como se ilustra na figura 132 e 133.

```
import jarray
import inspect
import os
from java.lang import Class
from java.lang import System
from java.sql import DriverManager, SQLException
from java.util.logging import Level
from java.util import ArrayList
from java.io import File
from org.sleuthkit.datamodel import SleuthkitCase
from org.sleuthkit.datamodel import AbstractFile
from org.sleuthkit.datamodel import ReadContentInputStream
from org.sleuthkit.datamodel import BlackboardArtifact
from org.sleuthkit.datamodel import BlackboardAttribute
```

Figura 132 - Bibliotecas visualização do ingest module

```
from org.sleuthkit.autopsy.ingest import IngestModule
from org.sleuthkit.autopsy.ingest.IngestModule import IngestModuleException
from org.sleuthkit.autopsy.ingest import DataSourceIngestModule
from org.sleuthkit.autopsy.ingest import IngestModuleFactoryAdapter
from org.sleuthkit.autopsy.ingest import IngestMessage
from org.sleuthkit.autopsy.ingest import IngestServices
from org.sleuthkit.autopsy.ingest import ModuleDataEvent
from org.sleuthkit.autopsy.coreutils import Logger
from org.sleuthkit.autopsy.casemodule import Case
from org.sleuthkit.autopsy.datamodel import ContentUtils
from org.sleuthkit.autopsy.casemodule.services import Services
from org.sleuthkit.autopsy.casemodule.services import FileManager
from org.sleuthkit.autopsy.casemodule.services import Blackboard
```

Figura 133 - Bibliotecas processamento do ingest module

A seguir é explicada a função de cada biblioteca usada no desenvolvimento dos dois ingest module, cada biblioteca tem uma função específica dentro do Autopsy:

- **'Jarray':** fornece suporte para arrays em Python.
- **'inspect'**: fornece recursos para introspecção de objetos.
- **'Os':** fornece recursos para interagir com o sistema operativo.
- **'java.lang.Class':** representa um objeto de classe em tempo de execução.
- **'java.lang.System':** fornece acesso às propriedades do sistema e métodos relacionados.
- **'java.sql.DriverManager':** fornece uma interface para efetuar a gestão JDBC.
- **'java.sql.SQLException':** indica um erro de SQL.
- **'java.util.logging.Level':** especifica o nível de registo para o logger.
- **'java.util.ArrayList':** implementa uma lista dinâmica.
- **'java.io.File':** representa um arquivo.
- **'org.sleuthkit.datamodel.SleuthkitCase':** representa um caso no Autopsy.
- **'org.sleuthkit.datamodel.AbstractFile'':** representa um arquivo no Autopsy.
- **'org.sleuthkit.datamodel.ReadContentInputStream':** fornece acesso a um fluxo de entrada para o conteúdo de um arquivo.
- **'org.sleuthkit.datamodel.BlackboardArtifact':** representa um artefato no blackboard.
- **'org.sleuthkit.datamodel.BlackboardAttribute':** representa um atributo no blackboard.
- **'org.sleuthkit.autopsy.ingest.IngestModule':** define a interface para um ingest module.
- **'org.sleuthkit.autopsy.ingest.DataSourceIngestModule':** define a interface para um ingest module que utiliza uma fonte de dados.
- **'org.sleuthkit.autopsy.ingest.IngestModuleFactoryAdapter':** fornece uma implementação padrão para a interface IngestModuleFactory.
- **'org.sleuthkit.autopsy.ingest.IngestMessage':** representa uma mensagem ingest.
- **'org.sleuthkit.autopsy.ingest.IngestServices':** fornece acesso aos serviços de ingest.
- **'org.sleuthkit.autopsy.ingest.ModuleDataEvent':** representa um evento de dados num ingest module.
- **'org.sleuthkit.autopsy.coreutils.Logger':** fornece um logger para o Autopsy.

- **‘org.sleuthkit.autopsy.casemodule.Case’:** representa um caso no Autopsy.
- **‘org.sleuthkit.autopsy.datamodel.ContentUtils’:** fornece vários métodos utilitários para trabalhar com o conteúdo de um caso no Autopsy.
- **‘org.sleuthkit.autopsy.casemodule.services.Services’:** fornece acesso aos serviços de um caso no Autopsy.
- **‘org.sleuthkit.autopsy.casemodule.services.FileManager’:** fornece um gestor de arquivos para o Autopsy.

O ingest module desenvolvido apresenta várias classes, mais precisamente:

- **’IviBmwDbIngestModuleFactory’;**
- **‘CicIviBmwDbIngestModule’.**

A classe **‘IviBmwDbIngestModuleFactory’** define o nome do módulo, descrição e a sua versão, como se ilustra na figura 134.

```
class CicIviBmwDbIngestModuleFactory(IngestModuleFactoryAdapter):

    moduleName = "Infotainment BMW CIC"

    def getModuleDisplayName(self):
        return self.moduleName

    def getModuleDescription(self):
        return "extract phone numbers,email,address,country,bluetooth," \
               "macaddress,devices were connected"

    def getModuleVersionNumber(self):
        return "1.0"

    def isDataSourceIngestModuleFactory(self):
        return True

    def createDataSourceIngestModule(self, ingestOptions):
        return CicIviBmwDbIngestModule()
```

Figura 134 - Informações do ingest module

A classe **‘CicIviBmwDbIngestModule’**, possui dois métodos relevantes para o funcionamento do módulo: ‘**startUp()’** e o **‘process()’**.

O método **‘startUp()’** é onde o módulo é inicializado. É um método que é chamado automaticamente pelo sistema quando o programa é iniciado e é geralmente usado para realizar a inicialização de recursos ou serviços necessários para o funcionamento do programa. É responsável por obter cada um dos parâmetros definidos na User Interface (UI) e que, por sua vez, foram armazenados nas *settings*, como se ilustra na figura 135.

```
def startUp(self, context):
    self.context = context
```

Figura 135 - Método startUp()

O método **'process()'** permite fazer a análise, sendo, onde se concentra grande parte do código do módulo. Este método é utilizado também para executar um programa ou comando num processo separado do processo principal do programa.

Resumidamente estes métodos **'startUp()', 'process()'**, permitem a possibilidade de apresentar ao utilizador, na interface Autopsy, o status de execução do módulo, para aplicar algum pré-processamento de tarefas necessárias para a análise do módulo (instalação e configuração), para definir a metodologia de análise e libertar recursos.

O Serviço *fileManager* permite localizar arquivos relevantes Para encontrar todos os arquivos de base de dados, usamos o método **'findFiles()'**. Este permite restringir a extensão dos ficheiros que serão obtidos. Por exemplo, colocando **'%.db'**, como parâmetro do método, este devolve ficheiros com extensão **.db**. Se não passar nenhum parâmetro, são devolvidos todos os ficheiros com qualquer extensão Este método devolve uma lista de objetos *AbstractFile. AbstractFile* dá acesso aos dados e conteúdo do ficheiro.

O objeto **'progressBar'** tem a função de criar uma barra de progresso no canto inferior direito e indica se o código está a funcionar devidamente, como se ilustra na figura 136.

```
def process(self, dataSource, progressBar):

    # we don't know how much work there is yet
    progressBar.switchToIndeterminate()

    # Use blackboard class to index blackboard artifacts for keyword s
    blackboard = Case.getCurrentCase().getServices().getBlackboard()

    # Find files named contacts.db, regardless of parent path
    fileManager = Case.getCurrentCase().getServices().getFileManager()
    files = fileManager.findFiles(dataSource, "contactbook_%.db")

    numFiles = len(files)
    progressBar.switchToDeterminate(numFiles)
    fileCount = 0
    for file in files:
```

Figura 136 - Método process()

A classe **'ContentUtils'**, permite guardar o ficheiro. Neste caso temos o caminho na pasta temporária do caso que estamos a analisar. Para evitar colisões de nomenclatura, nomeamos o arquivo, a seguir guarda o caminho do arquivo em **'lcIDbPath'**, conforme a figura 137.

```
lclDbPath = os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId()) + ".db")
ContentUtils.writeToFile(file, File(lclDbPath))
```

Figura 137 - Classe ContenUtils

Usamos o Java Database Connectivity (JDBC) para ligar a uma base de dados SQLite e executar consultas SQL. Para usar o JDBC. Em seguida, para criar uma ligação à base de dados usa uma instância da classe DriverManager, como consta na figura 138.

```
try:
    Class.forName("org.sqlite.JDBC").newInstance()
    dbConn = DriverManager.getConnection("jdbc:sqlite:%s" % lclDbPath)
```

Figura 138 - Java JDBC

5.1.2. Blackboard

O Blackboard permite que os módulos comuniquem e armazenem os resultados. O Blackboard é um conjunto de artefatos. Cada artefacto possui um conjunto de atributos. O Sleuth Kit tem muitos tipos de artefatos[8] e atributos[9] já definidos. Permite também criar blackboardartifact e blackboardAttribute.

BlackboardAttribute

Neste projeto foram usados vários atributos, como por exemplo:

- TSK_USER_ID,
- TSK_NAME,
- TSK_URL,
- TSK_ORGANIZATION,
- TSK_USER_ID,

[8] http://sleuthkit.org/sleuthkit/docs/jni-docs/4.6/enumorg_1_1sleuthkit_1_1datamodel_1_1_blackboard_artifact_1_1_a_r_t_i_f_a_c_t___t_y_p_e.html

[9] http://sleuthkit.org/sleuthkit/docs/jni-docs/4.6/classorg_1_1sleuthkit_1_1datamodel_1_1_blackboard_attribute.html

- TSK_NAME,
- TSK_PHONE_NUMBER,
- TSK_EMAIL,
- TSK_LOCATION,
- TSK_CITY,
- TSK_COUNTRY,
- TSK_PHONE_NUMBER,
- TSK_DATETIME,
- TSK_DATETIME_ACCESSED,
- TSK_DEVICE_ID,
- TSK_DESCRIPTION,
- TSK_TITLE,
- TSK_PATH,
- TSK_TEXT,
- TSK_VERSION.

Na figura 139, está uma parte do código para utilizar um atributo já definido, neste caso foi usado o atributo Contact_ID.

```
attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK_USER_ID.getTypeID(),
                                   IviBmwDbIngestModuleFactory.moduleName, Contact_ID))
```

Figura 139 - Código blackboardattribute definido

As figuras seguintes 140 e 141, apresentam o código para criar atributos, neste caso foi criado o atributo FamilyName.

```
family_name_att_type = blackboard.getOrAddAttributeType\
    ('BMW_FAMILY_NAME_TYPE',BlackboardAttribute.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "FamilyName")
```

Figura 140 - Código para criar um blackboardattribute novo

```
attributes.add(BlackboardAttribute(family_name_att_type,IviBmwDbIngestModuleFactory.moduleName, FamilyName))
```

Figura 141 - Código para criar um blackboardattribute novo

A figura 142, apresenta as colunas User ID, Name, FamilyName, Organization e URL, que fazem referência aos atributos escolhidos para analisar a base de dados contactbook_20120928.db.

Contacts

Table | Thumbnail | Summary

Source File	User ID	Name	FamilyName	Organization	URL
contactbook_20120928.db	794	Mark	Electrician		
contactbook_20120928.db	876	Mark		Extractor Fan...	
contactbook_20120928.db	877	Mark Stamp			
contactbook_20120928.db	878	Martin			
contactbook_20120928.db	879	Matt		RAC	
contactbook_20120928.db	565	Matthew	Fairey		
contactbook_20120928.db	582	Matthew	King	McDonald's C...	linkedin://#profile/369272906

Figura 142 - Blackboardattribute

BlackboardArtifact

Os artefactos do blackboard permitem que os módulos comuniquem entre si e permitem ainda exibir os dados como ilustra a figura 143. Por exemplo o artefacto TSK_CONTACT corresponde ao Results – Extracted Content "Contacts", que está associado a uma base de dados.

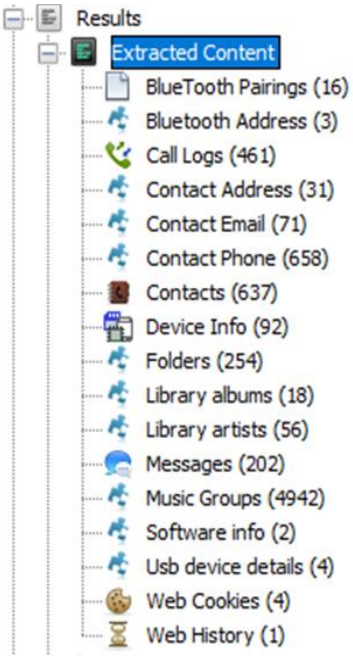


Figura 143 – Blackboardartifact

Neste projeto foram usados os seguintes artefactos disponibilizados pelo Autopsy:

- TSK_CALLLOG,
- TSK_CONTACT,
- TSK_MESSAGE,
- TSK_WEB_COOKIE,
- TSK_WEB_HISTORY,
- TSK_BLUETOOTH_PAIRING,
- TSK_WEB_HISTORY,
- TSK_DEVICE_INFO,

Na figura 144, está mencionado o código para usar um artefacto já disponível pelo Autopsy, neste exemplo foi usado o artefacto TSK_CONTACT.

```
art = file.newArtifact(BlackboardArtifact.ARTIFACT_TYPE.TSK_CONTACT)
```

Figura 144 - Código blackboardartifact já disponível no Autopsy

Neste projeto foram criados artefactos, como por exemplo:

- Bluetooth address,
- Contact Address,
- Contact email,
- Contact phone,
- Folders,
- Library albums,
- Library artists,
- Music groups,
- Software info,
- Usb device details.

Nas figuras 145 e 146, está mencionado o código para criar um artefacto, neste caso foi criado o artefacto Bluetooth Address.

```
arttId = blackboard.getOrAddArtifactType("TSK_BLUETOOTH_ADDRESS", "Bluetooth Address")
```

Figura 145 - Código para criar um blackboardartifact

```
art = file.newArtifact(arttId.getTypeID())
```

Figura 146 - Código para criar um blackboardartifact

5.1.3. Análise das bases de dados

Para analisar as bases de dados foi necessário estudar as bases de dados que constam na secção 4.10 capítulo 4, para definir quais as tabelas e colunas mais importantes para depois criar o ingest module. O ingest module ***IvibmwDataSourceIngestModule.py*** permite analisar diversas bases de dados, conforme se apresenta na tabela 36 e 37.

Tabela 36 - Bases de dados sistema IVI NBT serie 5 ano 2017

Base de dados Sistema IVI NBT marca BMW modelo serie 5 ano 2017	
Base de dados	**Autopsy Blackboardartifacts**
BrowserUrls	Web History
contactbook_20120928	Contacts
contactbook_20120928	Contact Phone
contactbook_20120928	Contact Email
contactbook_20120928	Contact Address
cookie	Web Cookies
contactbook_20120928	Bluetooth Address
f288251944.sqlite f287629400.sqlite f287269072.sqlite f287105230.sqlite	Messages
mme	Device Info
mme	Folders
mme	Usb device details
mme	Software info
mme	Music Groups
mme	Library albums
mme	Library artists
pim01	Bluetooth Pairings
pm8000102	Call Logs

Tabela 37 - Bases de dados sistema IVI NBT serie 5 ano 2017

Base de dados Sistema IVI NBT marca BMW modelo serie 7 ano 2017	
Base de dados	**Autopsy Blackboardartifacts**
BrowserUrls	Web History
contactbook_20120928	Contacts
contactbook_20120928	Contact Phone
contactbook_20120928	Contact Email
contactbook_20120928	Contact Address
contactbook_20120928	Bluetooth Address
mme	Device Info
mme	Folders
mme	Usb device details
mme	Software info
mme	Music Groups
mme	Library albums
mme	Library artists
pim01	Bluetooth Pairings
pm8000002 pm8000004	Call Logs

Para analisar as tabelas e colunas dessas bases de dados, foram criadas várias querys, para extrair os dados e importar os dados para a ferramenta Autopsy. A seguir são listadas as querys usadas para analisar as tabelas das diversas bases de dados.

Contacts

```
resultSet = stmt.executeQuery("SELECT contact_card_phone.Contact_ID, "
    "contact_card_phone.GivenName, contact_card_phone.FamilyName, "
    "contact_card_phone.Url, contact_card_phone.organisation"
    "FROM contact_card_phone ORDER BY contact_card_phone.GivenName")
```

Esta query seleciona dados da tabela "contact_card_phone", durante a consulta seleciona as colunas "Contact_ID", "GivenName", "FamilyName", "Url" e " organisation". Posteriormente ordena os resultados pela coluna "GivenName" em ordem crescente.

Contact Phone

```
resultSet = stmt.executeQuery("SELECT contact_card_phone.Contact_ID, "
        "contact_card_phone.GivenName, contact_card_phone.FamilyName,"
        "contact_card_phone.AdditionalName,contact_card_phone.Url, "
        "contact_card_phone.organisation,phone_data_phone.PhoneNumber"
```

```
        " FROM contact_card_phone JOIN phone_data_phone "
        "ON contact_card_phone.Contact_ID = phone_data_phone.Contact_ID "
        "ORDER BY contact_card_phone.GivenName")
```

Esta query seleciona informações de duas tabelas, "contact_card_phone" e "phone_data_phone", usando o operador JOIN. A tabela "contact_card_phone" contém informações sobre contatos, nome, sobrenome, ID de contato, enquanto a tabela "phone_data_phone" contém informações sobre números de telefone associados a esses contatos.

A query seleciona as colunas "Contact_ID", "GivenName", "FamilyName", "AdditionalName", "Url", "organisation" e "PhoneNumber" das duas tabelas. A seguir é usado o operador JOIN para combinar as duas tabelas, juntando as linhas onde o "Contact_ID" é igual em ambas as tabelas. Por fim, a query ordena os resultados por ordem crescente através da coluna "GivenName" na tabela "contact_card_phone".

Contact Email

```
resultSet = stmt.executeQuery("SELECT contact_card_phone.Contact_ID,"
     "contact_card_phone.GivenName, contact_card_phone.FamilyName,"
     "msg_data_phone.EmailAddr FROM contact_card_phone"
     "JOIN msg_data_phone ON"
     "contact_card_phone.Contact_ID = msg_data_phone.Contact_ID"
     "ORDER BY contact_card_phone.GivenName")
```

Esta query seleciona informações de duas tabelas, "contact_card_phone" e "msg_data_phone", usando o operador JOIN. A query seleciona as colunas "Contact_ID", "GivenName", "FamilyName" e "EmailAddr" das duas tabelas. A seguir usa o operador JOIN para combinar as duas tabelas, junta as linhas onde o "Contact_ID" é igual em ambas as tabelas. Por fim, a query ordena os resultados por ordem crescente através da coluna "GivenName" na tabela "contact_card_phone".

Contact Address

```
resultSet = stmt.executeQuery("SELECT contact_card_phone.Contact_ID,"
    "contact_card_phone.GivenName,"
    "contact_card_phone.FamilyName,"
    "contact_card_phone.AdditionalName, "
    "contact_card_phone.Url, contact_card_phone.organisation,"
    "address_phone.StreetHousenumber, address_phone.City,"
    "address_phone.Country, address_phone.Postalcode FROM "
    "contact_card_phone JOIN address_phone ON"
    "contact_card_phone.Contact_ID = address_phone.Contact_ID"
    "WHERE address_phone.crosssum > 0"
    "ORDER BY contact_card_phone.GivenName")
```

Esta query seleciona dados de duas tabelas, "contact_card_phone" e "address_phone", ambas estão ligadas pela coluna "Contact_ID". Durante a consulta filtra os resultados, selecionando apenas os registos em que a coluna "crosssum" na tabela "address_phone" é maior que 0. Finalmente, a consulta ordena os resultados pela coluna "GivenName" em ordem crescente.

Bluetooth Pairings

```
resultSet = stmt.executeQuery("SELECT Origin, BtAddress FROM bluetooth")
```

Esta query que seleciona as colunas "Origin" e "BtAddress" da tabela "bluetooth".

Bluetooth Address

```
resultSet = stmt.executeQuery("SELECT SID, INFO_KEY,"
    "INFO_VALUE FROM CE_DEVICE_INFO WHERE INFO_KEY ="
    "'IMEI' OR INFO_KEY = 'IMSI' OR INFO_KEY ="
    "'BluetoothAddress' or INFO_KEY = 'Model' ORDER BY SID")
```

Esta query seleciona os valores da coluna "SID", "INFO_KEY" e "INFO_VALUE" para as linhas onde a coluna "INFO_KEY" é igual a "IMEI", "IMSI", "BluetoothAddress" ou "Model". Posteriormente ordena os resultados por ordem crescente através da coluna "SID".

Call Logs

```
resultSet = stmt.executeQuery("SELECT CALLSTACKS.ID, CALLSTACKS.FN,"
    "CALLSTACKS.TEL_NR, STRFTIME('%s', CALLSTACKS.TIMESTAMP)"
    "AS TIMESTAMP FROM CALLSTACKS")
```

Esta query seleciona as colunas "ID", "FN", "TEL_NR", e usa a função "STRFTIME" aplicada à coluna "TIMESTAMP" da tabela "CALLSTACKS". A função "STRFTIME" é usada para formatar a data e hora armazenadas na coluna "TIMESTAMP" da tabela "CALLSTACKS". Especificamente, a função converte a data e hora para um formato Unix Timestamp, que é um número inteiro que representa o número de segundos desde 1º de janeiro de 1970 às 00:00:00 UTC.

Web History

```
resultSet = stmt.executeQuery("SELECT urls.id, urls.title,"
    "urls.url, visits.datevisit FROM urls LEFT JOIN visits")
```

Esta query seleciona as colunas "id", "title", "url" da tabela "urls" e a coluna "datevisit" da tabela "visits", e aplica um "LEFT JOIN" para unir as duas tabelas. Um "LEFT JOIN" retorna todos os registos da tabela à esquerda (no caso, "urls") e os registos correspondentes da tabela à direita (no caso, "visits"), se houver. Se não houver correspondência na tabela à direita, as colunas correspondentes na junção serão NULL.

Web Cookies

```
resultSet = stmt.executeQuery("SELECT cookies.name, cookies.host,"
                "cookies.path, cookies.lastAccessed FROM cookies")
```

Esta query seleciona as colunas "name", "host", "path" e "lastAccessed" da tabela "cookies".

Messages

```
resultSet = stmt.executeQuery("SELECT messages.id,"
    "messages.fromPhoneNumber,"
    "strftime("%s", substr(date,1,4) || "-" ||"
    "substr(date,5,2) || "-" ||"
    "substr(date,7,2) || "T" || substr(date,9,2) || ":" ||"
    "substr(date,11,2) || ":" || substr(date,13,2))"
    "as newdate, messages.subject FROM messages")
```

Esta query seleciona as colunas "id", "fromPhoneNumber", "newdate" e "subject" da tabela "messages". A coluna "newdate" é criada usando a função "strftime" para formatar a data e hora armazenadas na coluna "date" da tabela "messages" para o formato "YYYY-MM-DDTHH:MM:SS". Essa função usa “substr(string, start_position, length)” para extrair as diferentes partes da data e hora da coluna "date" e, em seguida, vai juntar essas substrings numa única string com o formato desejado.

O traço "-" é utilizado para separar as partes da data (ano, mês e dia) e o caractere "T" é utilizado para separar a data da hora.

Device Info

```
resultSet = stmt.executeQuery("SELECT msid, lastseen,"
    "mssname, name, identifier, mountpath FROM mediastores")
```

Esta query seleciona as colunas "msid", "lastseen", "mssname", "name", "identifier" e "mountpath" da tabela "mediastores".

Folders

```
resultSet = stmt.executeQuery("SELECT foldername,"
                "last_sync, basepath FROM folders")
```

Esta query seleciona as colunas "foldername", "last_sync" e "basepath" da tabela "folders".

Usb device details

```
resultSet = stmt.executeQuery("SELECT deviceserialno,"
                    "lastseen FROM usbdevicedetails")
```

Esta query seleciona as colunas "deviceserialno" e "lastseen" da tabela "usbdevicedetails".

Software info

```
resultSet = stmt.executeQuery("SELECT software_info.version"
                              "FROM software_info")
```

Esta query seleciona a coluna "version" da tabela "software_info".

Music Groups

```
resultSet = stmt.executeQuery("SELECT categorydata_custom.name "
                              "FROM categorydata_custom")
```

Esta query seleciona a coluna "name" da tabela "categorydata_custom".

Library albums

```
resultSet = stmt.executeQuery("SELECT library_albums.album"
                              " FROM library_albums")
```

Esta query seleciona a coluna "album" da tabela "library_albums".

Library artists

```
resultSet = stmt.executeQuery("SELECT library_artists.artist"
                              "FROM library_artists")
```

Essa query SQL seleciona a coluna "artist" da tabela "library_artists".

O ingest module ***CicIvibmwDataSourceIngestModule.py*** permite analisar diversas bases de dados, conforme se apresenta na tabela 38 e 39.

Tabela 38 - Bases de dados sistema IVI CIC serie 3 ano 2012

Base de dados Sistema IVI CIC marca BMW modelo serie 3 ano 2012	
Base de dados	**Autopsy Blackboard artifacts**
contactbook_20101214	Contacts
contactbook_20101214	Contact Phone
contactbook_20101214	Contact Email
contactbook_20101214	Contact Address
contactbook_20101214	Bluetooth Pairings
mme	Device Info
mme_library	Folders
mme_custom	Software info
mme_library	Library albums
mme_library	Library artists

Tabela 39 - Bases de dados sistema IVI CIC serie 3 ano 2010

Base de dados Sistema IVI CIC marca BMW modelo serie 3 ano 2010	
Base de dados	**Autopsy Blackboard artifacts**
contactbook_20101214	Contacts
mme	Device Info
mme_library	Folders
mme_custom	Software info
mme_library	Library albums
mme_library	Library artists

Para analisar as tabelas e colunas das bases de dados, foram criadas várias querys, para encontrar os dados e apresentar os dados na ferramenta Autopsy. A seguir são listadas as query que foram usadas.

Contacts

```
resultSet = stmt.executeQuery("SELECT contact_card_phone.Contact_ID,"
    "contact_card_phone.GivenName, contact_card_phone.FamilyName,"
    "contact_card_phone.organisation FROM contact_card_phone"
    "ORDER BY contact_card_phone.GivenName")
```

Contact Phone

```
resultSet = stmt.executeQuery("SELECT contact_card_phone.Contact_ID,"
     "contact_card_phone.GivenName,contact_card_phone.FamilyName,"
     "phone_data_phone.PhoneNumber FROM contact_card_phone "
     "JOIN phone_data_phone ON contact_card_phone.Contact_ID ="
     "phone_data_phone.Contact_ID "
     "ORDER BY contact_card_phone.GivenName")
```

Contact Email

```
resultSet = stmt.executeQuery("SELECT contact_card_phone.Contact_ID,"
     "contact_card_phone.GivenName,contact_card_phone.FamilyName,"
     "msg_data_phone.EmailAddr FROM contact_card_phone "
     "JOIN msg_data_phone ON contact_card_phone.Contact_ID ="
     "msg_data_phone.Contact_ID "
     "ORDER BY contact_card_phone.GivenName")
```

Contact Address

```
  resultSet = stmt.executeQuery("SELECT contact_card_phone.Contact_ID,"
        "contact_card_phone.GivenName,contact_card_phone.FamilyName,"
        "address_phone.StreetHousenumber, address_phone.City,"
        "address_phone.Country, address_phone.Postalcode FROM"
        "contact_card_phone JOIN address_phone ON"
        "contact_card_phone.Contact_ID = address_phone.Contact_ID"
        "WHERE address_phone.crosssum > 0"
        "ORDER BY contact_card_phone.GivenName")
```

Bluetooth Pairings

```
resultSet = stmt.executeQuery("SELECT Bt_ID, HEX(Bt_address)"
                              "as hexBt from bluetooth")
```

Esta query seleciona as colunas "Bt_ID" e "Bt_address" da tabela "bluetooth" e também converte a coluna "Bt_address" para a representação hexadecimal usando a função "HEX()". O resultado final é uma lista de registos que contém o valor da coluna "Bt_ID" e o valor hexadecimal da coluna "Bt_address".

Device Info

```
resultSet = stmt.executeQuery("SELECT msid, capabilities,"
    "mssname, name, identifier, mountpath FROM mediastores")
```

Folders

```
resultSet = stmt.executeQuery("SELECT foldername,"
             "last_sync, basepath FROM folders")
```

Software info

```
resultSet = stmt.executeQuery("SELECT version FROM software_info")
```

Library albums

```
resultSet = stmt.executeQuery("SELECT library_albums.album"
                              "FROM library_albums")
```

Library artists

```
resultSet = stmt.executeQuery("SELECT library_artists.artist"
                              "FROM library_artists")
```

6. Resultados

6.1. Análise com ingest module desenvolvido

Neste projeto vamos usar os dois plugins ingest modules desenvolvidos neste projeto, o ingest module ***Infotainment BMW NBT*** para o sistema IVI NBT e o ingest module ***Infotainment BMW CIC*** para o sistema IVI CIC.

O objetivo destes ingest module é permitir localizar, analisar e extrair informação das bases de dados dos sistemas IVI NBT e CIC e apresentar esses dados de forma intuitiva ao investigador.

Para executar os ingest modules que foram criados neste projeto, temos de selecionar Tools – Run Ingest Modules, de seguida aparecem vários ingest modules e escolhemos os ingest modules que foram desenvolvidos neste projeto como se ilustra na figura 147.

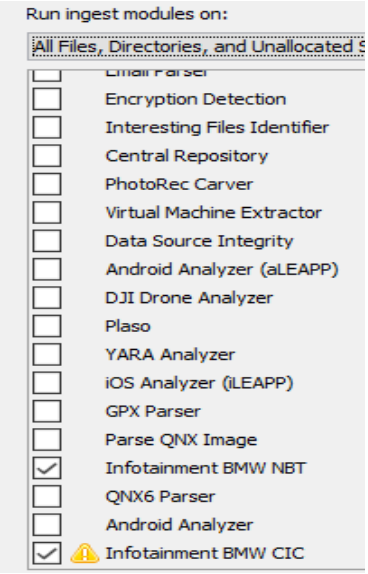


Figura 147 - Executar ingest module

6.2. Resultados

Depois de executar o ingest module adequado ao sistema IVI, foram encontrados diversos dados com muita informação relevante para uma investigação. Nas secções seguintes vão ser apresentados os resultados das aquisições forenses realizadas nos sistemas IVI CIC e NBT. Salienta-se que foram analisados quatro discos rígidos de quatro sistemas IVI de veículos reais.

6.2.1. Resultados do Sistema IVI NBT ano 2017 modelo serie 5

Ao executar o ingest module *Infotainment BMW NBT*, foi encontrada uma série de dados, como demonstra na figura 148 e 149.

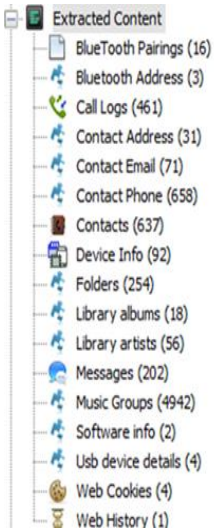


Figura 148 - Results-Extracted Content

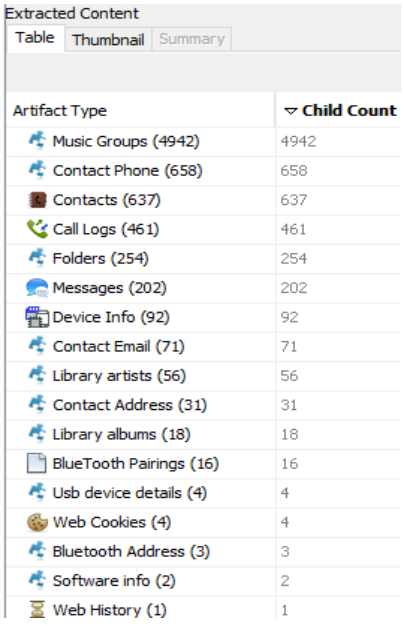


Figura 149 - Listing Extract Content

A tabela 40, apresenta as bases de dados que corresponde a cada *Results - Extracted Content* e que tipo de dados essas bases de dados possuem.

Tabela 40 - Results - Extracted Content

Base de dados	Results - Extracted Content	Tipo de dados
BrowserUrls	Web History	Histórico da Web
cookie	Web Cookies	Cookies
contactbook_20120928	Contacts	Nome e apelido
contactbook_20120928	Contact Phone	N.º telefone, nome e apelido
contactbook_20120928	Contact Email	E-mail, nome e apelido
contactbook_20120928	Contact Address	Nome, apelido, morada, cidade, código-postal e pais
f288251944.sqlite f287629400.sqlite f287269072.sqlite f287105230.sqlite	Messages	Registo de mensagens, data/hora do envio da mensagem, n.º telefone e o texto da mensagem
mme	Device Info	Registo da sincronização de vários dispositivos como smartphones, CD/DVDs, iPods e pendrives com o sistema IVI
mme	Folders	Demonstra o nome das pastas, o caminho da pasta e última sincronização da pasta
mme	Usb device details	Informação dos dispositivos USB, que estiveram ligados ao sistema IVI
mme	Software info	Informação do software
mme	Music Groups	Grupos musicais
mme	Library albums	Lista de álbuns musicais
mme	Library artists	Lista de músicas
pim01	Bluetooth Pairings	Bluetooth mac address, IMEI, IMSI, modelo do telefone e marca
pm8000102 pm8000096 pm8000108	Call Logs	Registo de chamadas, n.º de telefone e hora/data da chamada

Desta análise podemos verificar que este veículo possui uma grande quantidade de dados, nas próximas figuras são apresentados os dados que foram encontrados.

Contacts como se ilustra na figura 150.

ource File	User ID	Name	FamilyName	Organization	URL
contactbook_20120928.db	628	Aznav	van	Московский государст	www.vkontakte
contactbook_20120928.db	560	Aziz			www.plamingo.
contactbook_20120928.db	603	Miste		Alliance of Beauty	www.capris.cor
contactbook_20120928.db	737	Bank			www.bochk.cor
contactbook_20120928.db	581	Boris	kovsky	Multiple Companies	linkedin://#prof
contactbook_20120928.db	615	Justin			linkedin://#prof
contactbook_20120928.db	571	Stefa	Gitman	AIESEC	linkedin://#prof
contactbook_20120928.db	588	Wei J		KPMG Singapore	linkedin://#prof
contactbook_20120928.db	552	Anar	ov		linkedin://#prof
contactbook_20120928.db	617	Allen			linkedin://#prof
contactbook_20120928.db	575	Arjan	a	Multiple Companies	linkedin://#prof
contactbook_20120928.db	583	Tanit		RA	linkedin://#prof
contactbook_20120928.db	554	Yuche			linkedin://#prof
contactbook_20120928.db	555	wassi	er	Technische Universit	linkedin://#prof
contactbook_20120928.db	608	Eduar	v	Multiple Companies	linkedin://#prof
contactbook_20120928.db	582	Matth		McDonald's Corporati	linkedin://#prof
contactbook_20120928.db	382	Андр	нов	Multiple Companies	linkedin://#prof

Figura 150 - Dados Contacts

Contact Phone, como se ilustra na figura 151.

Source File	User ID	Name	FamilyName	Phone Number
contactbook_20120928.db	496	Alex	Невский	+372
contactbook_20120928.db	468	Але	уренко	+380
contactbook_20120928.db	361		омер	+447
contactbook_20120928.db	439	Сен	сников	+380
contactbook_20120928.db	424	Евге	ов	+380
contactbook_20120928.db	417	Еле	вецкая	+380
contactbook_20120928.db	382	Анд	монов	+380
contactbook_20120928.db	699		sh	07910
contactbook_20120928.db	698			07922
contactbook_20120928.db	697	Jasc		13910

Figura 151 - Dados Contact Phone

Contact Email, como se ilustra na figura 152.

Source File	User ID	Name	FamilyName	Email
contactbook_20120928.db	813			luhanuk@
contactbook_20120928.db	584	Aaron		aaron_ku
contactbook_20120928.db	551	Alexa	kov	alexande
contactbook_20120928.db	617	Allen		allen9809
contactbook_20120928.db	552	Anar	nov	anarbayr
contactbook_20120928.db	607	Anua	ev	yuu_78@
contactbook_20120928.db	575	Arjan	la	arjanjaiye
contactbook_20120928.db	560	Aziz		curic.a@p
contactbook_20120928.db	586	Barna	nery	blassonne
contactbook_20120928.db	581	Boris	rkovsky	boris@ma
contactbook_20120928.db	623	Brend		adnerb31
contactbook_20120928.db	594	Chels	son	chelsea@
contactbook_20120928.db	622	Danie	ov	D.Y.Sozo

Figura 152 - Dados Contact Email

Contact Address, como se ilustra na figura 153.

Source File	User ID	Name	FamilyName	Location	City	PostalCode	Country
contactbook_20120928.db	627	Ahr	yev	Вильямс	essa	ppe	
contactbook_20120928.db	423	Ene		21 Pitfie	don	SE2	а Британія
contactbook_20120928.db	759	Wil	ng	11 Chur	ingbourne	ME	Kingdom
contactbook_20120928.db	731	Ant		Margare	ds	LS8	Kingdom
contactbook_20120928.db	733	Aur		10 Robi	rogate	HG	Kingdom
contactbook_20120928.db	548	Kar	andiyarova		Escaldes	AD	a
contactbook_20120928.db	547		le Inc.	1 Infinite	pertino	950	States
contactbook_20120928.db	620	Har	tic	Korica H	canica	753	and Herzegovina
contactbook_20120928.db	593	And	anov	17 Kuzn	goveshchensk	675	
contactbook_20120928.db	609	Ira	vdiva	Киевска	v	073	
contactbook_20120928.db	549	Dur	s	Antalya	alya	070	
contactbook_20120928.db	578	Osm	alci	Istanbul	anbul	000	
contactbook_20120928.db	550	Seb	amwegha	Uganda	npala	+25	a
contactbook_20120928.db	940	Ant	rpe	Margare			
contactbook_20120928.db	607	Anu	ayev		rborne		Kingdom

Figura 153 - Contact Address

Bluetooth Address e ***Bluetooth Pairings***, como se ilustra nas figuras 154 e 155.

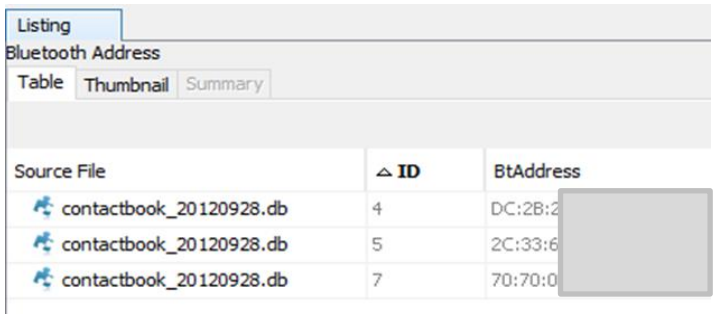


Figura 154 - Bluetooth Address

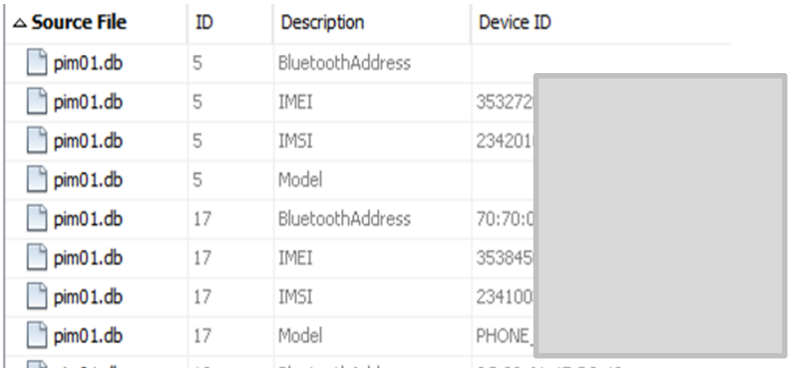


Figura 155 - Bluetooth Pairings

Call logs, como se ilustra na figura 156.

Figura 156 - Call Logs

Massages, como se ilustra na figura 157.

Source File	ID	fromPhoneNumber	Text	Date/Time
f287105230.sqlite	237	+447	know we got it there chillin	2017-06-24 09:26:07 BST
f287105230.sqlite	238	+447	yim we got crack there	2017-06-24 09:24:34 BST
f287105230.sqlite	239	+447	car	2017-06-24 09:18:07 BST
f287105230.sqlite	240	+447		2017-06-24 08:20:48 BST
f287105230.sqlite	241	+447		2017-06-24 06:09:45 BST
f287105230.sqlite	242	+447		2017-06-24 05:44:58 BST
f287105230.sqlite	243	+447	know man	2017-06-24 05:44:14 BST
f287105230.sqlite	244	+447	down	2017-06-24 05:39:21 BST
f287105230.sqlite	245	+447		2017-06-24 05:39:08 BST
f287105230.sqlite	246	+447		2017-06-24 05:38:53 BST
f287105230.sqlite	247	+447		2017-06-24 05:35:41 BST
f287105230.sqlite	248	+447		2017-06-24 05:35:30 BST
f287105230.sqlite	249	+447		2017-06-24 05:21:49 BST
f287105230.sqlite	250	+447		2017-06-24 05:15:18 BST
f287105230.sqlite	251	+447	will just pass the fuck out j	2017-06-24 05:15:14 BST
f287105230.sqlite	252	+447		2017-06-24 05:13:56 BST
f287105230.sqlite	253	+447	guy?	2017-06-24 03:51:14 BST

Figura 157 - Messages

Web Cookies, como se ilustra na figura 158.

Listing
Web Cookies
Table | Thumbnail | Summary

Source File	Name	URL	Path	Date Accessed
cookie.db	VEHICLEAUTH	b2v.bmwgroup.de	/DUMMY_PATH	2017-03-12 14:00:21 GMT
cookie.db	JSESSIONID	b2v.bmwgroup.de	/DUMMY_PATH	2017-03-12 14:00:21 GMT
cookie.db	VEHICLEAUTH	cngw.bmwgroup.de	/DUMMY_PATH	2017-03-12 14:00:21 GMT
cookie.db	JSESSIONID	cngw.bmwgroup.de	/DUMMY_PATH	2017-03-12 14:00:21 GMT

Figura 158 - Web Cookies

Web History, como se ilustra na figura 159.

Web History
Table | Thumbnail | Summary

Source File	ID	Title	URL	Date Accessed
BrowserUrls.db	1	http://www.google.com/	http://www.google.com/	0000-00-00 00:00:00

Figura 159 - Web History

Device Info, como se ilustra na figura 160.

Source File	ID	Date/Time	Description	Name	Device ID	Path
mme	1	2017-07-02 11:21:39 BST	internet	SocketDMBSlave	/dev/socket	/dev/socket
mme	2	2017-07-02 11:21:40 BST	dmb	DMB-Radio	/dev/tuner/dmb	/dev/tuner/dmb
mme	3	2017-07-02 11:21:39 BST	devb	Voicenotes	B74B8D051CDD8E5D3B8DE0253417EF8C	/mnt/quota/mm/Voicenotes
mme	4	2017-07-02 11:21:39 BST	devb	HardDisk	EAB1B3AD4E68ABFA8DD22B22BA147FE4	/mnt/quota/mm/EntServer
mme	5	2017-07-02 11:21:39 BST	devb	OnlineEntertainment	FECDBD5E6993CCD6B66DFF43653C0887	/mnt/quota/mm/OnlineEntertainment/Content
mme	6	2014-01-17 11:01:56 GMT	mediafs2wire	Charlotte's iPhone	88:53:95:B8:19:33-1	
mme	7	2014-01-17 11:01:56 GMT	ipod	Charlotte's iPhone	DQGJK12UDTC0	
mme	8	2014-06-05 17:05:09 BST	mediafs2wire	C2105	B4:52:7D:D4:19:28	
mme	9	2014-03-11 19:04:44 GMT	mediafs2wire	Esmee's iPhone	90:B9:31:81:C9:25-2	
mme	10	2014-07-22 13:58:14 BST	ipod	Esmee's iPhone	DNPLMDLDFFGC	
mme	11	2014-07-22 13:58:15 BST	mediafs2wire	Esmee's iPhone	90:B9:31:81:C9:25-1	
mme	12	2014-03-18 17:39:34 GMT	mediafs2wire	Esmee's iPhone	90:B9:31:81:C9:25	
mme	13	2014-06-22 11:41:19 BST	audiocd	CD/DVD	ce12501f	
mme	14	2014-12-30 11:31:47 GMT	audiocd	WHAT HAVE WE BECOME	8d0aa50c	

Figura 160 - Device Info

Music Groups, como se ilustra na figura 161.

Source File	Name
mme	Cheryl Cole
mme	Chris Brown
mme	Chyi Chin
mme	City of London Sinfonia, Andrew Watkinson, Nicholas Ward & Eddie
mme	Coldplay
mme	Conor Maynard
mme	Craig Colton

Figura 161 - Music Groups

Library Albums, como se ilustra na figura 162.

Source File	album
mme	
mme	Capo (Deluxe Edition)
mme	Farruko Edition
mme	Jim Jones-Ghost Of Rich Porter
mme	Overpowered
mme	Pray IV Reign
mme	Soul Togetherness 2015
mme	The Kitchen
mme	WHAT HAVE WE BECOME
mme	
mme	Capo (Deluxe Edition)
mme	Farruko Edition

Figura 162 - Library Albums

Folders, como se ilustra na figura 163.

Source File	foldername	Path	Date Modified
mme		/	2017-06-11 02:22:11 BST
mme	Suntex	/Suntex/	2017-06-11 02:22:11 BST
mme	temp cads for china	/temp cads for china/	2017-06-11 02:22:11 BST
mme	Yes Master	/Yes Master/	2017-06-11 02:22:11 BST
mme	photos from China	/photos from China/	2017-06-11 02:22:11 BST
mme	a s erotic	/a s erotic/	2017-06-11 02:22:11 BST
mme	beautiful bottoms	/beautiful bottoms/	2017-06-11 02:22:11 BST
mme	AS Sophie	/AS Sophie/	2017-06-11 02:22:11 BST
mme	EROTIC ELLIE	/EROTIC ELLIE/	2017-06-11 02:22:11 BST
mme	China Photos April 2012	/China Photos April 2012/	2017-06-11 02:22:11 BST
mme	Trends Hud 2	/Trends Hud 2/	2017-06-11 02:22:11 BST
mme	photos	/photos/	2017-06-11 02:22:11 BST
mme	Farruko Edition - Imperio Nazza (2013)	/Farruko Edition - Imperio Nazza (2013)/	2017-06-11 02:22:11 BST
mme	Jim Jones - Pray IV Rain - 2009	/Jim Jones - Pray IV Rain - 2009/	2017-06-11 02:22:11 BST
mme	Jim Jones - The Kitchen (2016) [MP3~320Kbps]~[Hunter] [FRG]	/Jim Jones - The Kitchen (2016) [MP3~320Kbps]~[Hunter] [FRG]/	2017-06-11 02:22:11 BST

Figura 163 – Folders

Usb devide details, como se ilustra na figura 164.

Source File	deviceserialno
mme	F2LS93GWHFYH
mme	07AA140886E835AD

Figura 164 - Usb devide details

Software info, como se ilustra na figura 165.

Source File	Version
mme	NBT_B13322A

Figura 165 - Software info

6.2.2. Resultados do Sistema IVI NBT ano 2017 modelo serie 7

Ao executar o ingest module Infotainment BMW NBT, foi encontrada uma série de dados como demonstra na figura 166 e 167.

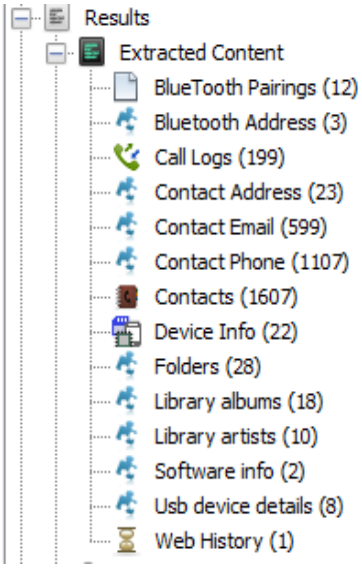


Figura 166 - Results - Extracted Content

Listing
Extracted Content
Table | Thumbnail | Summary

Artifact Type	Child Count
Contacts (1607)	1607
Contact Phone (1107)	1107
Contact Email (599)	599
Call Logs (199)	199
Folders (28)	28
Contact Address (23)	23
Device Info (22)	22
Library albums (18)	18
BlueTooth Pairings (12)	12
Library artists (10)	10
Usb device details (8)	8
Bluetooth Address (3)	3
Software info (2)	2
Web History (1)	1

Figura 167 - Listing Extracted Content

A tabela 41, apresenta as bases de dados que corresponde a cada *result - extracted content*, e que tipo de dados essas bases de dados possuem.

Tabela 41 - Results extracted content

Base de dados	Results – Extracted Content	Tipos de dados
BrowserUrls	Web History	Histórico da Web
contactbook_20120928	Contacts	Nome e apelido
contactbook_20120928	Contact Phone	N.º telefone, nome e apelido
contactbook_20120928	Contact Email	E-mail nome e apelido
contactbook_20120928	Contact Address	N.º telefone, nome e apelido
mme	Device Info	Registo da sincronização de vários dispositivos como smartphones, CD/DVDs, iPods e pendrives com o sistema IVI
mme	Folders	Demonstra o nome das pastas, como o seu caminho, e última sincronização da pasta
mme	Usb device details	Informação dos dispositivos USB, que estiveram ligados ao sistema IVI
mme	Software info	Informação do software
mme	Library albums	Lista de álbuns musicais
mme	Library artists	Lista de músicas/artistas
pim01	Bluetooth Pairings	Bluetooth mac address, IMEI, IMSI, modelo do telefone, marca
pm8000002 pm8000004	Call Logs	Registo de chamadas, n.º de telefone e hora/data da chamada

Desta análise podemos verificar que este veículo possui uma grande quantidade de dados, nas próximas figuras são apresentados os dados que foram encontrados.

Contacts como se ilustra na figura 168.

Source File	User ID	Name	FamilyName	Organization	URL
contactbook_20120928.db	2749	Taj	asović	ja	
contactbook_20120928.db	78	Gor			
contactbook_20120928.db	1427	Gor			
contactbook_20120928.db	2837	Livi	včić	farmacija PC	
contactbook_20120928.db	2619	Ra	adar		
contactbook_20120928.db	2563		r	Marul	
contactbook_20120928.db	2586		e		
contactbook_20120928.db	2554	Kris			
contactbook_20120928.db	2584	Ma	šić	- Bobovik I	
contactbook_20120928.db	2570		bović	na Viru (Ž	
contactbook_20120928.db	59	Dju	ncic		
contactbook_20120928.db	2391	Filip	cević	ko	

Figura 168 - Dados Contacts

Contact Phone, como se ilustra na figura 169.

Source File	Use...	Name	FamilyName	Phone Number	Data Source
contactbook_20120928.db	1	Vo		**211	LogicalFileSet1
contactbook_20120928.db	10	An	gic	091 5	LogicalFileSet1
contactbook_20120928.db	100	Ivi	bisa	091 5	LogicalFileSet1
contactbook_20120928.db	101	Ivi	čić	098 3	LogicalFileSet1
contactbook_20120928.db	102	Ivi	ovic	091 2	LogicalFileSet1
contactbook_20120928.db	103	Ivi		(29) 4	LogicalFileSet1
contactbook_20120928.db	104	Ivi	ić	098 1	LogicalFileSet1
contactbook_20120928.db	105	Ivn	ić	091 5	LogicalFileSet1
contactbook_20120928.db	106	Ivo	unovic	098 1	LogicalFileSet1
contactbook_20120928.db	107	Jas	c	632	LogicalFileSet1
contactbook_20120928.db	108	Jas	ć	+385	LogicalFileSet1
contactbook_20120928.db	109	Jas	ka	(1) 17	LogicalFileSet1
contactbook_20120928.db	11	Ap		098 6	LogicalFileSet1
contactbook_20120928.db	110	Jas	kalj	(21) 5	LogicalFileSet1
contactbook_20120928.db	111	Jos	o	+385	LogicalFileSet1
contactbook_20120928.db	112	Jos	sevic	+385	LogicalFileSet1
contactbook_20120928.db	113	Jos	usinovic	+385	LogicalFileSet1
contactbook_20120928.db	114	Jos	usinovic	(1) 11	LogicalFileSet1
contactbook_20120928.db	115	Jos	usinovic-...	098 3	LogicalFileSet1

Figura 169 - Dados Contact Phone

Contact Email, como se ilustra na figura 170.

Contact Email

Table | Thumbnail | Summary

Source File	User ID	Name	FamilyName	Email
contactbook_20120928.db	2318	Lid	arić	lidija power.hr
contactbook_20120928.db	2385	Ala	ec	alan
contactbook_20120928.db	2313	Če	ko	zelj harma.hr
contactbook_20120928.db	2230	Bru	rinić	brur com
contactbook_20120928.db	2229	Da	er	dam g.hr
contactbook_20120928.db	2227	An	jić	antu m
contactbook_20120928.db	2497	Kri	ić	ksor
contactbook_20120928.db	2220	Že	ć	zsoi
contactbook_20120928.db	2216	Ro	ek	robe
contactbook_20120928.db	2215	Ma	unović	mari
contactbook_20120928.db	2472	Ma	ović	Marl hr
contactbook_20120928.db	2213	Div	ić	Divr
contactbook_20120928.db	2214	Pe	ić	peta r

Figura 170 - Dados Contact Email

Contact Address, como se ilustra na figura 171.

Contact Address

Table | Thumbnail | Summary

Source File	Use...	Name	FamilyName	Location	City	PostalCode	Country
contactbook_20120928.db	2261	Zl	čić	turčić	Zagre	040	Hrvatska
contactbook_20120928.db	1822	Va	cin	mate	zagre	040	
contactbook_20120928.db	2302	Iv	unić	bahnh	luterb	42	ch
contactbook_20120928.db	2577	Ro	ć	Vilima			Hrvatska
contactbook_20120928.db	1812	Mi	lici	Via Fr	Novat	0026	Italy
contactbook_20120928.db	2656	Sa	dić	Ul.dr.	SOLIN		
contactbook_20120928.db	2762	Ut	z	Serifa	Soylesi Umran	775	Turska
contactbook_20120928.db	2135	IG	DIN	Ratka	Rijeka	000	Hrvatska
contactbook_20120928.db	2111	Al	arić	Ozalj	Zagre	000	Croatia
contactbook_20120928.db	2739	To	ć	Ozalj	Zagre	000	Croatia
contactbook_20120928.db	2141	Br	oševic	Konra	offenb	073	Deutchalnd
contactbook_20120928.db	1902	Kr	ć	J.P.Ka	Rijeka	000	Hrvatska
contactbook_20120928.db	2739	To	ć	Grade	Gornji	255	Hrvatska
contactbook_20120928.db	1557	Br	ković	Goren	Gornji	255	CROATIA

Figura 171 - Dados Contact Address

Bluetooth Address, como se ilustra na figura 172 e 173.

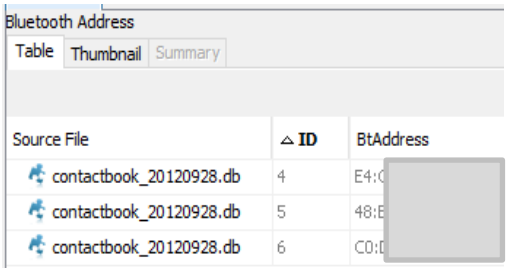


Figura 172 - Dados Bluetooth Address

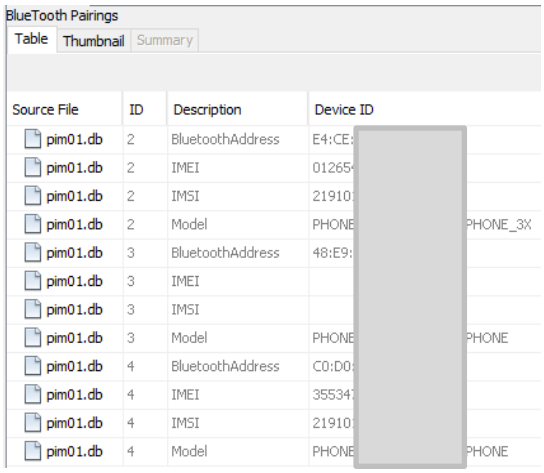


Figura 173 – Dados Bluetooth Pairings

***Call logs*,** como se ilustra na figura 174.

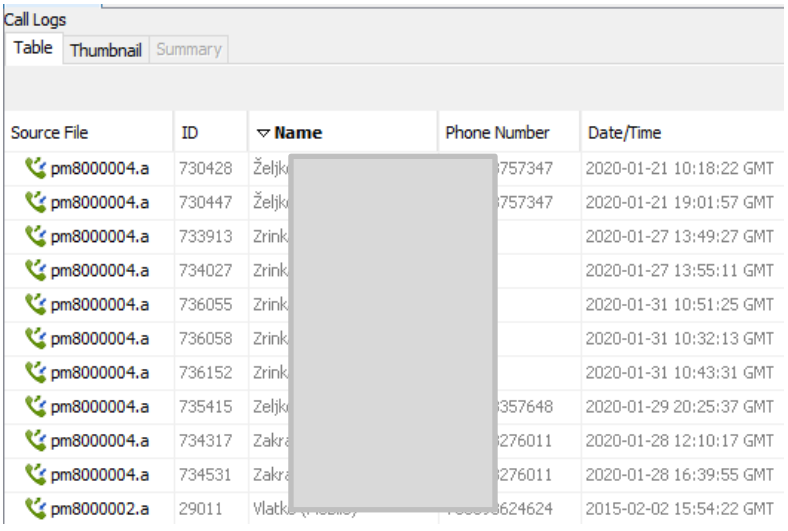


Figura 174 - Dados Call logs

Web History, como se ilustra na figura 175.

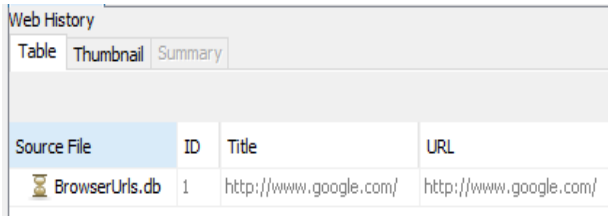


Figura 175 - Dados Web History

Device Info, como se ilustra na figura 176.

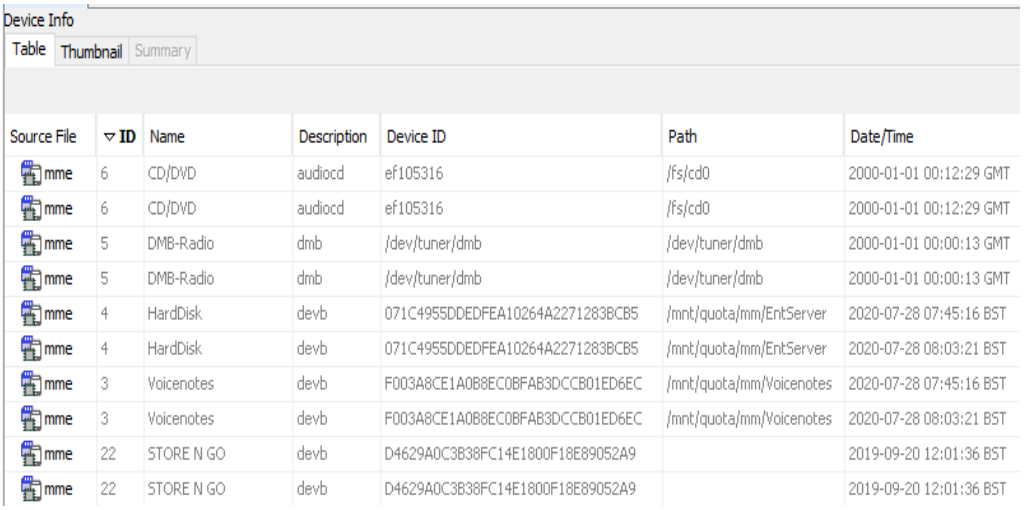


Figura 176 - Device Info

Folders, como se ilustra na figura 177.

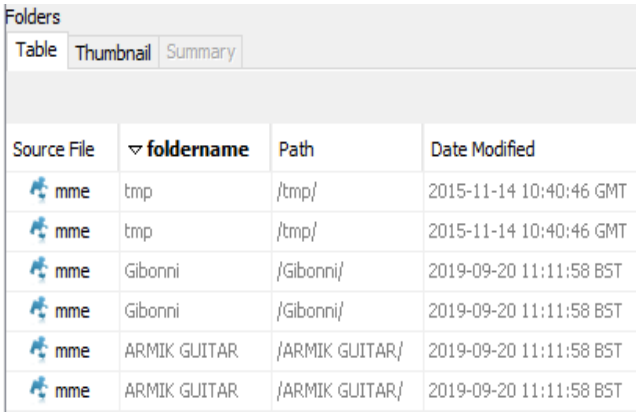


Figura 177 – Folders

Library Albums, como se ilustra na figura 178.

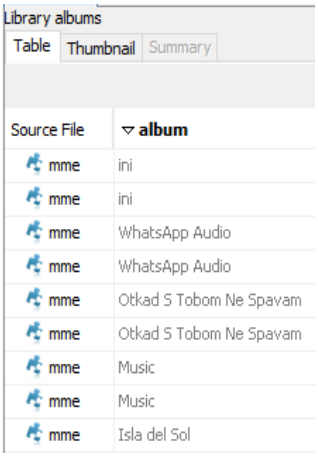


Figura 178 - Library Albums

Library artists, como se ilustra na figura 179.

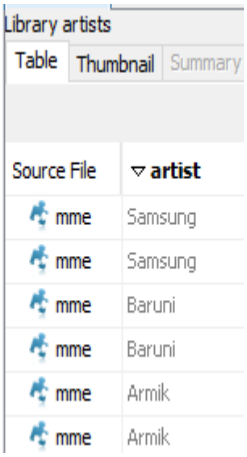


Figura 179 - Library artists

Software info, como se ilustra na figura 180.

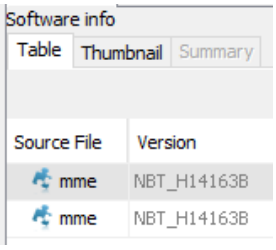


Figura 180 - Software info

Usb devide details, como se ilustra na figura 181.

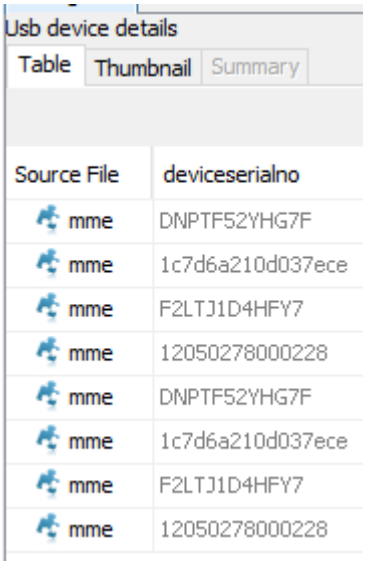


Figura 181 - Usb devide details

6.2.3. Resultados do Sistema IVI CIC ano 2012 modelo série 3

Ao executar o ingest module Infotainment BMW CIC, foi encontrada uma série de dados como demonstra na figura 182 e 183.

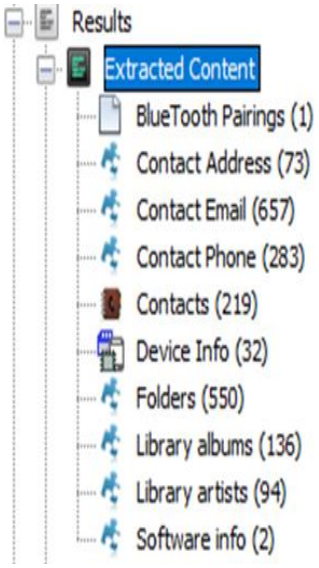


Figura 182 - Results - Extracted Content

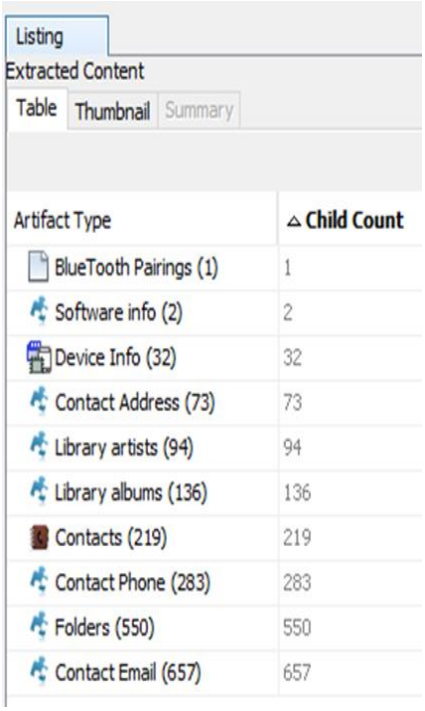


Figura 183 - Listing Extracted Content

A tabela 42, apresenta as bases de dados a que corresponde a cada *results - extracted content* e que tipo de dados essas bases de dados possuem.

Tabela 42 - Results - Extracted Contente CIC 2012

Base de dados	Results – Extracted Content	Tipo de dados
contactbook_20101214	Contacts	Nome e apelido
contactbook_20101214	Contact Phone	N.º telefone, nome e apelido
contactbook_20101214	Contact Email	E-mail nome e apelido
contactbook_20101214	Contact Address	Nome, apelido, morada, cidade, código-postal e pais
contactbook_20101214	Bluetooth Pairings	Bluetooth mac address
mme	Device Info	Registo da sincronização de vários dispositivos como smartphones, CD/DVDs, iPods e pendrives com o sistema IVI
mme_library	Folders	Demonstra o nome das pastas, o caminho da pasta, e última sincronização da pasta
mme_custom	Software info	Informação do software
mme_library	Library albums	Lista de albuns musicals
mme_library	Library artists	Lista de musicas/artistas

Contacts, como se ilustra na 184.

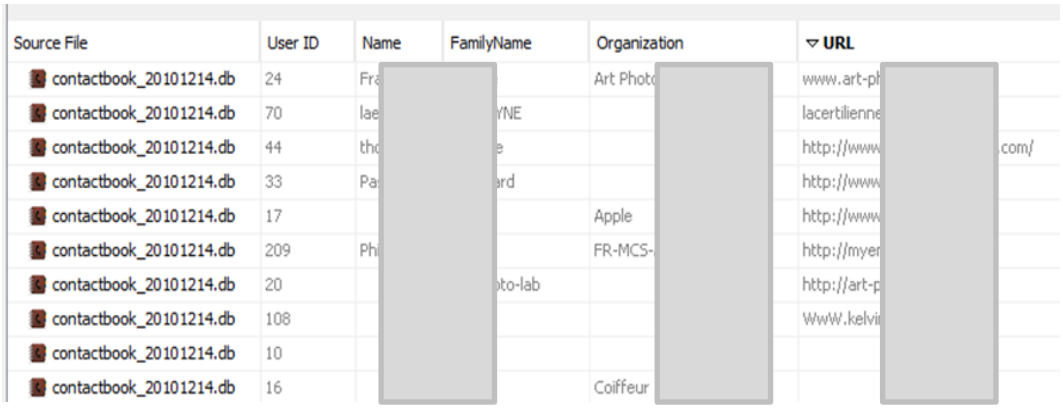


Figura 184 - Dados Contacts

Contact Phone, como se ilustra na figura 185.

Source File	User ID	Name	FamilyName	Phone Number
contactbook_20101214.db	122	celin	nano	0617
contactbook_20101214.db	119	Mare	ovitch	0678
contactbook_20101214.db	68	laure	vera	0676
contactbook_20101214.db	50	Laur	nut	0681
contactbook_20101214.db	25	Oph	ujean	0658
contactbook_20101214.db	219	Alain	chetta	+331
contactbook_20101214.db	219	Alain	chetta	+331
contactbook_20101214.db	219	Alain	chetta	+336
contactbook_20101214.db	218	Dian	ardi	0666
contactbook_20101214.db	217	Vince	lin	+334
contactbook_20101214.db	217	Vince	lin	+336
contactbook_20101214.db	216	Mari	elm	0630
contactbook_20101214.db	215	Laet	maux	0616
contactbook_20101214.db	214	Fabi	n	0682
contactbook_20101214.db	212	Elvir		0625

Figura 185 - Dados Contact Phone

Contact Email, como se ilustra na figura 186.

Source File	User ID	Name	FamilyName	Email
contactbook_20101214.db	207	Amél	ébert	zephy mail.com
contactbook_20101214.db	178	Hu	ao	yuqia om
contactbook_20101214.db	142	Caroli	nin	yuesh ail.com
contactbook_20101214.db	53	Yichur	en	yichur com
contactbook_20101214.db	216	Marie	helm	xtatix
contactbook_20101214.db	217	Vincer	ttlin	vwittli
contactbook_20101214.db	186	Violair	ndu	violair il.com
contactbook_20101214.db	137	Véron	ry	veron com
contactbook_20101214.db	64	Ariel	sta	tsila.9
contactbook_20101214.db	44	thoma	rrage	thoma rrage.com
contactbook_20101214.db	44	thoma	rrage	thoma nail.com
contactbook_20101214.db	98	Thierr	ton	thierry osoft.com
contactbook_20101214.db	49	Thierr	antier	thierry marchy.com
contactbook_20101214.db	130	Sydne	rtinie	sydne nail.com
contactbook_20101214.db	43	Sue	meron	sue.c rchy.com
contactbook_20101214.db	9	Suaer	ault	suaen ail.com
contactbook_20101214.db	171	Steph	ERI	steph .lu
contactbook_20101214.db	188	Sebas	ignant	sroign m
contactbook_20101214.db	179	Sophi	ANQUIN	sophie racle.com
contactbook_20101214.db	69	Sophi	Charry	sodec .fr
contactbook_20101214.db	21	Stéph	tomobiles	sfranc el.fr

Figura 186 - Dados Contact Email

Contact Address, como se ilustra na figura 187.

User ID	Name	FamilyName	Location	City	PostalCode	Country
170	olivi	ERI	9b R	Brin	9126	France
170	olivi	ERI	14 Ru	Tas	9160	France
148	mon	YRET	1 Pla	Villa		France
169	miso	ERI	17, r	Calu	9300	France
86	Viola	ben	La Pe	Can	2800	France
162	Viola	RRIN	2 B a	TAS	9160	FRANCE
13	Vale	zonne	25 ru	onage 69330 France		France
72	Tiph	LAUNAY	1 che	VAL	9670	
151	Thar	uyen	18 av	y Buyer	Lyo	9009 France
21	Stép	tomobiles	6, ru	LYO	9009	
171	Step	ERI	61, r	Lux	329	Luxembourg
208	Soni	SSOT	8 rue	LYO	9009	
67	Sara	MBERLAND	438 R	ix	DIV	1220 FRANCE
106	Sala	mel	Rue M	Tas	9160	France
106	Sala	mel	14 Ru	peret	Tas	9160 France
47	Rom	ambodut	112 r	Lyo	9006	France
101	Rem	ntieu	38b r	cha		France

Figura 187 - Dados Contact Address

Device Info, como se ilustra na figura 188.

Listing
Device Info
Table | Thumbnail | Summary

Source File	ID	Description	Name	Device ID	Path
mme	1	devb	IBA	44AA25C0172FED4D6A2CA2EEA7E55DC8	/mnt/hbdata/IBA
mme	2	devb	HardDrive	C6F0E7EF9B361692BDEB926ADA5BCA5F	/mnt/hbmedia/entertainmentserver/
mme	3	audiocd	CD/DVD	ef105316	
mme	4	devb	KINGSTON	ED0903F288AB6705770B777EBFF4CEB6	/fs/usb0
mme	5	devb	KINGSTON	FD8F421B0C7149EFA51AEB9E80070E9F	/fs/usb0
mme	6	devb	Neu	C717F2E106F32D82344189EC8A72E213	
mme	7	dvdvideo	DVD	D137AA09E4D218BFB9DBE9ABD5C01251	
mme	8	devb	Transcend	91EEA68819F9180BC9F712FA66198990	
mme	9	devb	MyDisc	12B7AAFD2C150276115784D029114FBE	
mme	10	audiocd	CD/DVD	9411e80a	
mme	11	audiocd	Absolute Oriental	cd11400f	
mme	12	audiocd	Kalamak Ya Habibi	47072e06	
mme	13	devb	Rock mars 2012	26E01FE8A74589E96248D04FBEC4C6E1	
mme	14	devb	USB	189A280257F945F382D4562568FD1812	
mme	15	devb	Lexar	940C13DA6C53FEBCAFA4F259139DDEFE	
mme	16	devb	USB	02DCD8ED6B7B42484C76765BD7E4155B	

Figura 188 - Device info

Folders, como se ilustra na figura 189.

Listing
Folders
Table | Thumbnail | Summary

Source File	foldername	Path
mme_library		/
mme_library	Foo Fighters	/Foo Fighters/
mme_library	Muse	/Muse/
mme_library	Nirvana	/Nirvana/
mme_library	Shaka Ponk	/Shaka Ponk/
mme_library	Foo Fighters	/Foo Fighters/Foo Fighters/
mme_library	Absolution	/Muse/Absolution/
mme_library	Black Hole and Revelat...	/Muse/Black Hole and Revelations/
mme_library	The Resistance	/Muse/The Resistance/
mme_library	Supermassive Black Hole	/Muse/Supermassive Black Hole/
mme_library	The 2nd Law	/Muse/The 2nd Law/
mme_library	MTV Unplugged	/Nirvana/MTV Unplugged/
mme_library	Nevermind	/Nirvana/Nevermind/
mme_library	Bad Porn Movie Trax	/Shaka Ponk/Bad Porn Movie Trax/
mme_library	The Geeks and the Jer...	/Shaka Ponk/The Geeks and the Jerkin'Socks/

Figura 189 - Folders

Library Albums, como se ilustra na figura 190.

Source File	album
mme_library	The 2nd Law
mme_library	The Resistance
mme_library	MTV Unplugged in New York
mme_library	Nevermind
mme_library	Bad Porn Movie Trax
mme_library	The Geeks and the Jerkin' Socks
mme_library	Franz Ferdinand
mme_library	Tonight
mme_library	You Could Have It So Much Better
mme_library	Hands All Over (Deluxe Version)
mme_library	It Won't Be Soon Before Long
mme_library	Misery - Single
mme_library	Moves Like Jagger (feat. Christina Aguilera) - Single
mme_library	Overexposed
mme_library	Hotel
mme_library	I Like to Score
mme_library	Play
mme_library	We are all made of stars

Figura 190 - Library Albums

Library artists, como se ilustra na figura 191.

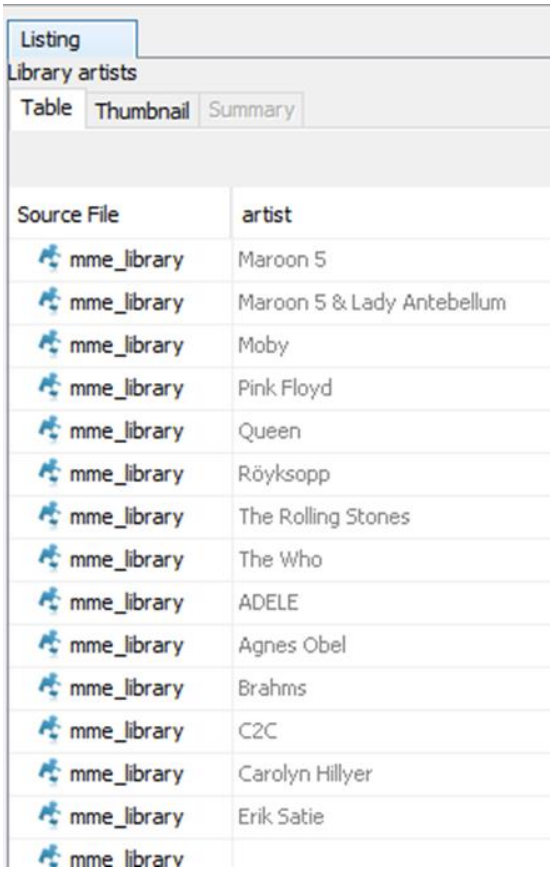


Figura 191 - Library artists

Software info, como se ilustra na figura 192.

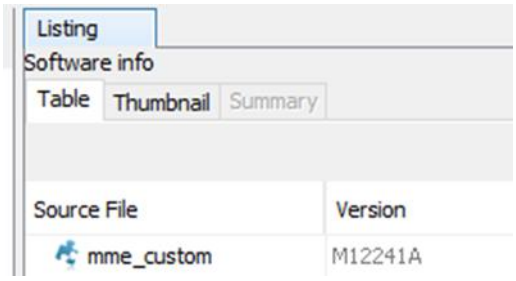


Figura 192 - Software info

6.2.4. Resultados do Sistema IVI CIC ano 2010 modelo série 3

Ao executar o ingest module Infotainment BMW CIC, foi encontrada uma série de dados como demonstra na figura 193 e 194.

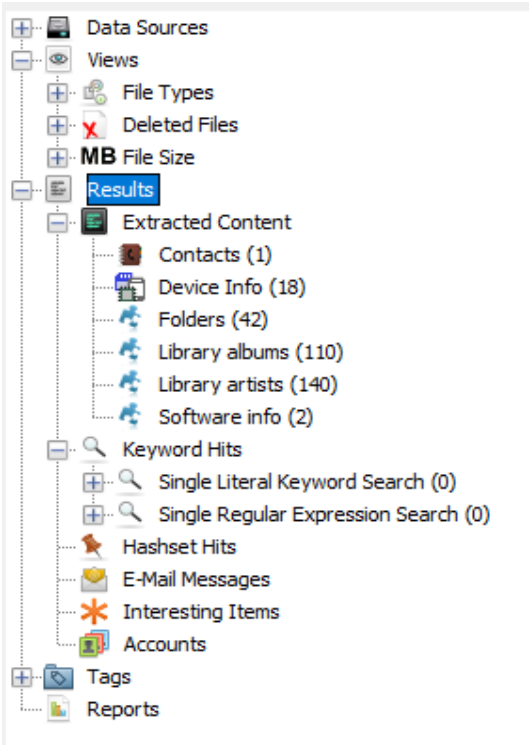


Figura 193 - Results - Extracted Content

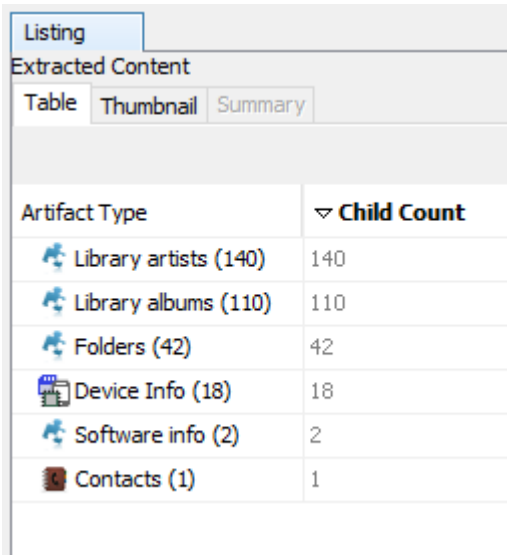


Figura 194 - Listing - Extracted Content

A tabela 43, apresenta as bases de dados a que corresponde a cada *Results - Extracted Content* e que tipo de dados essas bases de dados possuem.

Tabela 43 - Results Extracted Content

Base de dados	Results – Extracted Content	Tipos de dados
mme	Device Info	Registo da sincronização de vários dispositivos como smartphones, CD/DVDs, iPods e pendrives com o sistema IVI
mme_library	Folders	Demonstra o nome das pastas, o caminho da pasta e a última sincronização da pasta
mme_custom	Software info	Informação do software
mme_library	Library albums	Lista de álbuns musicais
mme_library	Library artists	Lista de álbuns musicais

Device Info, como se ilustra na figura 195.

Listing
Device Info
Table | Thumbnail | Summary

Source File	ID	Name	Description	Device ID	Path
mme	1	IBA	devb	277B34B0DDC2A043F5F2978717DA7BDD	/mnt/hbdata/IBA
mme	2	HardDrive	devb	6D2179F0BDA90BBD3124E557E6CA49CC	/mnt/hbmedia/entertainmentserver/
mme	3	CD/DVD	audiocd	6d097b0a	
mme	4	The Joshua Tree	audiocd	8a0bc50b	/fs/cd0
mme	5	MTV Ao Vivo	audiocd	eb11cb12	
mme	6	CD/DVD	audiocd	ad08c70c	
mme	7	Varios	devb	F66B8DB06B941D3746FB2D5D4B872050	
mme	8	Meus Arquivos MP	devb	88DB9B8B61FF6E36DC28461D47982381	
mme	9	Meus Arquivos MP	devb	D9D0D2DCEDB5AFD5FE5258CA4AD04457	
mme	10	Meus Arquivos MP	devb	B5CF897C5D52F360EC19898932E42B42	
mme	11	Meus Arquivos MP	devb	5619A04A6871CBE4A90F9B4F56032B84	
mme	12	Os Meus Ficheiro	devb	3A7393F7F2D041EFFC71954F2094A963	
mme	13	Disc	devb	163673ABA8E55498871F8A9CA6D2BA55	
mme	14	Disc	devb	559987205AF8D22803FEB20F3B42AF9F	
mme	15	Disc	devb	B46E9C2CF8BDB54E284F97060D1240B2	
mme	16	Disc	devb	8D13A954EEF66C27A52208A065A96C92	
mme	17	Disc	devb	0E567B2060858DCC6F9620C8641A442D	
mme	18	Disc	devb	EF0C87204BFE8C0F53C8110C1E232F01	/fs/cd0

Figura 195 - Device Info

Folders, como se ilustra na figura 196.

Listing
Folders
Table | Thumbnail | Summary

Source File	foldername	Path
mme_library	swe2	/swe2/
mme_library	common.zip	/swe2/common.zip/
mme_library	anim	/swe2/common.zip/anim/
mme_library		/
mme_library		/

Figura 196 - Folders

Library Albums, como se ilustra na figura 197.

Source File	album
mme_library	Eragon OST
mme_library	Ballo fuori tempo
mme_library	UP_by Profo [LT]
mme_library	All The Lost Souls
mme_library	Classificados
mme_library	Sou metade sem ti
mme_library	Páginas da Vida - Nacional
mme_library	Um e o Outro
mme_library	Hit Parade '97
mme_library	Tudo o que Você Queria Ouvir
mme_library	All the Lost Souls
mme_library	One Way
mme_library	Pablo Alborán - En Acústico
mme_library	Viva La Vida or Death And All His Friends
mme_library	Grenade - Single
mme_library	Privateering
mme_library	Wake Me Up - Single
mme_library	Wings - Single
mme_library	When I Was Your Man - Single

Figura 197 - Library Albums

Library artists, como se ilustra na figura 198.

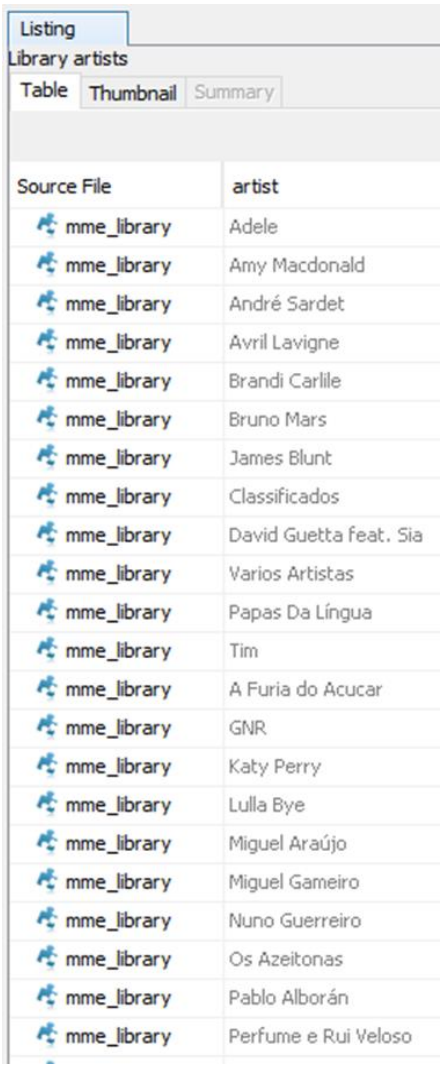


Figura 198 - Library artists

Software info, como se ilustra na figura 199.

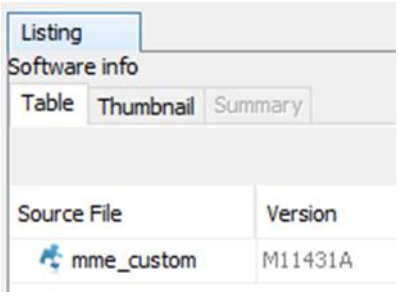


Figura 199 - Software info

6.3.Discussão de resultados

A tabela 44, menciona o número total de dados que foram encontrados nos diversos sistemas IVI.

Tabela 44 - Tipo e número de dados encontrados nos sistemas IVI CIC e NBT

Tipo de dados encontrados	Sistema IVI NBT serie 3 ano 2017	Sistema IVI NBT serie7 ano 2017	Sistema IVI CIC serie3 ano 2012	Sistema IVI CIC serie3 ano 2010
Contacts	637	1607	219	1
Contact Phone	658	1107	283	0
Contact Email	71	599	657	0
Contact Address	31	23	73	0
Bluetooth Pairings	3	3	1	0
Call Logs	461	199	Não possui	Não possui
Messages	202	0	Não possui	0
Device Info	92	22	32	18
Folders	254	28	550	42
Usb device details	4	8	Não possui	Não possui
Software Info	2	2	2	2
Music Groups	4942	0	Não possui	Não possui
Library albums	18	18	136	110
Library artists	56	10	94	140
Web Cookies	4	0	Não possui	Não possui
Web History	1	1	Não possui	Não possui

Dos resultados da tabela, podemos tirar algumas ilações. O sistema IVI NBT fornece mais dados e informações que o sistema IVI CIC. O sistema IVI CIC, é antigo, a nível tecnológico é inferior, logo tem menos dados e funcionalidades do que o sistema IVI NBT. O sistema IVI CIC não possui nenhuma base de dados relativa a Call Logs, Messages, Usb device details, Music Groups, Web Cookies e Web History.

O sistema IVI CIC do modelo de 2010 não possui nenhum Bluetooth Address, porque não foi sincronizado nenhum dispositivo a esse sistema IVI. Relativamente ao Bluetooth Address dos outros sistemas IVI, verifica-se que foram sincronizados dispositivos ao sistema IVI.

Relativamente aos dados Contacts, verifica-se uma grande quantidade de dados nos sistemas IVI NBT e no sistema IVI CIC do ano 2012. O sistema IVI CIC do ano de 2010 possui um contacto. O grande volume de contactos está relacionado com o número de contatos de cada telefone que sincronizou com o sistema IVI.

Os sistemas IVI NBT possuem uma grande quantidade de dados relativo a Call Logs, os sistemas IVI CIC, não possuem nenhuma base de dados referente a Call Logs.

No sistema IVI NBT do veículo série 5 do ano 2017 possui Messages, enquanto o modelo série 7 do ano 2017 não possui nenhum dado referente a Messages. Os sistemas IVI CIC não possuem nenhuma base de dados relativo a Messages.

Ambos os sistemas IVI NBT e CIC possuem dados Library albums e Library artists, é de realçar que os sistemas CIC possuem mais dados, devido ao seu utilizador usar mais esta funcionalidade no referido sistema IVI.

Em relação aos dados Music Groups, só o modelo NBT série 5 do ano 2017 possui dados. O modelo NBT série 7 ano 2017 possui esta base de dados, mas não tinha qualquer informação. Comparativamente ao sistema IVI CIC estes não possuem estas base de dados, logo não guardam nenhuma informação relativa a Music Groups.

No que respeita aos dados Web Cookies e Web History, só o modelo NBT possui estes dados. Os sistemas IVI CIC não possuem estas base de dados, logo não guardam nenhuma informação relativa a Web Cookies e Web History.

No que concerne aos dados Software Info, Folders e Device Info, ambos os sistemas IVI possuem dados, quanto aos dados Usb device details estes só estão presentes nos sistemas IVI NBT, o sistema IVI CIC, não possui nenhuma base de dados relativamente a Usb device details.

Em comparação entre os sistemas IVI NBT e CIC, o sistema IVI NBT é mais completo, possui mais base de dados e mais funcionalidades que o sistema IVI CIC, logo, os sistemas IVI NBT fornecem mais provas, mas, ambos os sistemas são valiosos para uma investigação, os dois contêm provas relevantes e em quantidades variadas.

7. Conclusão

Neste projeto podem ser retiradas várias conclusões sobre o tema abordado, assim como ideias e sugestões para trabalhos futuros.

7.1. Conclusões

A análise forense a sistemas IVI é um campo emergente na análise forense digital que permite aos investigadores encontrar uma grande quantidade de dados importantes para investigações forenses.

Torna-se difícil obter informações e encontrar os dados existentes nos sistemas IVI, em virtude das diferenças de hardware, sistemas operativos e devido a serem sistemas proprietários. Existe pouca informação sobre o software usado, e as aquisições podem ser limitadas. Outra dificuldade é aceder ao disco rígido HDD, temos de remover o sistema IVI. Para desmontar é uma tarefa bastante complexa, esses sistemas IVI estão num local difícil de acesso, tem de se desmontar uma parte do tablier para efetuar a sua remoção. A documentação existente sobre análise forense de sistemas IVI é muito limitada.

Este projeto teve como objetivo identificar quais os dados que podem ser encontrados nos sistemas IVI NBT e CIC da marca BMW e quais os dados que podem ser úteis para uma investigação.

Investigou-se quais os dados presentes no modelo CIC série 3 do ano de 2010 e 2012 e no modelo NBT série 5 e série 7 do ano 2017, apurou-se que os dados relevantes estão em ficheiros de tipo base de dados SQLite. Foram detetadas 12 bases de dados no sistema IVI NBT e 5 bases de dados no sistema IVI CIC. Nestas bases de dados encontraram-se dados tais como:

- Contatos;
- Bluetooth mac address;
- IMEI;
- IMSI;
- Modelo do telefone;
- Registos de chamadas;

- Mensagens SMS;
- Registo e informação de vários dispositivos (smartphone, pendrives, ipod, CD/DVD);
- Lista de grupos musicais;
- lista de músicas;
- Histórico da Web;
- Cookies.

Foram desenvolvidos dois plugins ingest module para a ferramenta Autopsy que permitiram extrair os dados mencionados anteriormente de forma automática. Primeiro foi necessária uma extração manual dos dados devido ao Autopsy não suportar nativamente sistemas de ficheiros QNX. Depois de extrair os dados manualmente, foram importados para o Autopsy para efetuar uma análise com os ingest modules que foram criados.

Durante a análise foi possível validar os dois plugins que foram executados e implementados com sucesso, tendo encontrado automaticamente informação dos vários discos rígidos. Em particular encontraram-se os dados mencionados anteriormente.

Ao comparar os dados recuperados entre os quatro sistemas IVI analisados, é de realçar que dois sistemas IVI NBT têm mais dados que o modelo CIC. Os modelos CIC são mais antigos e a nível tecnológico inferiores, logo tem menos funcionalidades. O sistema IVI CIC não possui dados de registo de chamadas, mensagens SMS, IMEI, IMSI, modelo do telefone, usb device details, cookies e histórico da Web. O sistema IVI NBT é mais completo, possui mais base de dados e mais funcionalidade que o sistema IVI CIC, logo, o sistema IVI NBT oferece mais provas, mas, ambos os sistemas são valiosos para uma investigação, os dois contêm provas relevantes e em quantidades variadas.

O trabalho que foi desenvolvido neste projeto abre caminho para os investigadores forenses, permitindo saber que tipo de informação podem encontrar e quais os dados mais importantes inseridos nos sistemas IVI da marca BMW.

Os dados armazenados nestes sistemas IVI estão a aumentar. No futuro esses sistemas ainda vão fornecer mais dados aos investigadores. Em resumo este trabalho fornece resultados a serem considerados para os preparativos estratégicos de futuras investigações forenses no campo de análise forense digital aos veículos nomeadamente a sistemas IVI. É importante desenvolver plugins ingest module para a ferramenta Autopsy, para permitir analisar os

sistemas IVI, uma vez que, a ferramenta Berla é paga e é a única existente no mercado para fazer análises aos sistemas IVI. Em suma, neste sistema IVI reside muito potencial, a pesquisa neste campo é importante e mais investigação deve ser feita nesta área.

Inicialmente, foram definidos um conjunto de objetivos, os quais foram cumpridos, posso dizer que no desenvolvimento deste projeto, as aprendizagens foram enormes e estou feliz com os resultados alcançados, assim como, com a aprendizagem pessoal.

7.2. Trabalhos futuros

Os veículos vão continuar a possuir sistemas IVI e equipados com o sistema operativo QNX Nuetrino RTOS, este sistema operativo ainda permanecerá bem representado na análise forense nos sistemas IVI.

Existem vários dados que não foram encontrados neste projeto, nomeadamente dados referentes à navegação GPS.

Como o Autopsy não suporta nativamente sistemas de ficheiros QNX, é necessário o desenvolvimento e implementação de acesso nativo a sistemas de ficheiros QNX na ferramenta Autopsy.

Vários veículos possuem a sua própria aplicação, como a aplicação da BMW "MyBmw connecteddrive", existe uma grande troca de dados entre o veículo e a aplicação mobile, essas aplicações podem ser importantes para a análise forense digital.

No mercado dos sistemas IVI, foi implementado um novo sistema operativo Android Automotive, as evidências contidas nesse sistema ainda são desconhecidas, é um sistema operativo que deve ser explorado e analisado.

A análise forense digital a sistemas IVI, terá de lidar com novos desafios, como tecnologia de condução autónoma e novos tipos de ameaças à cibersegurança dos sistemas IVI, incluindo ataques aos sistemas IVI. Neste caso é importante um estudo da segurança e privacidade dos sistemas IVI, incluindo a identificação de vulnerabilidades e investigação de possíveis violações de privacidade, para estes sistemas não sofrerem nenhum ataque. Além da tecnologia de condução autónoma tem um potencial enorme para gerar grandes quantidades de dados sobre atividade do veículo, incluindo os dados de sensores e vídeo. A análise desses dados pode ser útil em caso de acidentes de viação envolvendo veículos autónomos e para a investigação de crimes.

No futuro, devido ao elevado número de tecnologias inseridas nos veículos, provavelmente haverá um aumento de crimes informáticos direcionados a esses veículos.

À medida que a tecnologia avança e novos desafios surgem, é importante para os investigadores de análise forense digital continuar a explorar novas técnicas e ferramentas para garantir a precisão e integridade das provas recolhidas.

Referências Bibliográficas

[1] J. Cabrera, "Marco de referencia para el desarrollo de un análises forense a fuentes de evidencias digitales en sistemas infotainment de vehiclos," Abril 2018.

[2] J. Sammons, The Basics of Digital Forensics: The Primer for Getting Started in Digital Forensics, 2.ª ed., Syngress, 2012.

[3] C. Whelan, J. Sammons, B. Mcmanus e T. Fenger, "Retrieval of infotainment system artifacts from vehicles using Ive," *Journal of applied digital evidence,* vol. 1, 2018.

[4] BERLA, *Aplicação da Berla iVe mobile,* 2022.

[5] E. Bates, "The APB - Digital Vehicle Forensics," vol. 24, nº 2, 2018.

[6] SWGDE, "Best practices for vehicle infotainnment and telematics systems," 13 01 2022.

[7] J. Andrew e D. Whatson, Digital Forensics Processing and Procedures, 2013.

[8] "A Road Map for Digital Forensic Research. Technical report, DFRWS," 2021.

[9] M. Antunes e B. Rodrigues, Introdução à Cibersegurança a internet, os aspetos legais e a análise digital forense, FCA, 2018.

[10] J. Abad, "Computer Forensic - Automatización con Autopsy," p. 140, 2018.

[11] S. Ferreira, "Machine learning based digital forensics application to detect tampered multimedia files," 2021.

[12] F. Marques, "Digital Forensics Procedures For Apple Devices," 2017.

[13] Norwich University , "5 Steps for conducting computer forensics investigations," 11 09 2017.

[14] L. Xiaodong, Introdutory computer forensics, Springer, 2018.

[15] S. Bommisetty, R. Tamma e H. Mahalik, Pratical Mobile Forensics, Pack Publishing, 2014.

[16] "Investigation of JTAG and ISP Techniques for Forensic Procedures," 26 11 2021.

[17] "What is JTAG," [Online]. Available: https://www.datarecovery.co.za/faq/what-is-jtag.html. [Acedido em 23 10 2022].

[18] C. M. Silveira, "Metodologia para reconstrução de dados em dispositivos móveis com sistema operacional android mediante aplicação de técnicas ISP e combination firmware," 2020.

[19] C. Silveira, R. Sousa, R. Albuquerque, G. Nze, G. Júnior, A. Orozco e L. Villalba, "Methodology for Forensics Data Reconstruction on Mobile Devices with Android Operating System Applying In-System Programming and Combination Firmware," *Multidisciplinary Digital Publishing Institute,* 20 06 2020.

[20] K. Kent, T. Grance e D. Hung, "Guide to integrating forensic techniques into incident response - recommendations of the National Institute of Standards and technology," 2006.

[21] J. Lacroix, "Vehicular infotainment forensics collecting data and putting it into perspective," 2017.

[22] C. Smith, The car hackers handbook, 2016.

[23] "BMW SÉRIE 3," 17 03 2017. [Online]. Available: https://www.blogauto.com.br/wp-content/2017/03/BMW-S%C3%A9rie-3-Gran-Turismo-2018-12.jpg. [Acedido em 12 08 2022].

[24] M. Stogsdill e w. Bortles, "Introduction to forensic passenger acquisition data from vehicle infotaining telematics systems," 01 01 2017.

[25] R. Atkinson, C. Urquhart e X. Bellekens, "Cyber - Security internals of a Skoda Octavia VRS," 2016.

[26] "What does the computer in a car do?," 2023. [Online]. Available: https://auto.howstuffworks.com/under-the-hood/trends-innovations/question113.htm. [Acedido em 25 01 2023].

[27] U. o. Warwick, "How dashcams help and hinder forensics," 11 6 2020. [Online]. Available: https://phys.org/news/2020-06-dashcams-hinder-forensics.html. [Acedido em 05 01 2022].

[28] Berla, "Products," [Online]. Available: https://berla.co/ecosystem/. [Acedido em 06 06 2022].

[29] F. Spissu, "Android infotainment system - a prototype based on Android Automotive," 2018.

[30] "Blackberry QNX," 2022. [Online]. Available: https://blackberry.qnx.com/en/company. [Acedido em 02 02 2022].

[31] "Blackberry QNX," 2022. [Online]. Available: https://blackberry.qnx.com/en/industries/connected-autonomous-vehicles. [Acedido em 02 02 2022].

[32] B. QNX, "QNX Neutrino Real-Time Operating System (RTOS)," 2022. [Online]. Available: https://blackberry.qnx.com/en/products/foundation-software/qnx-rtos. [Acedido em 02 02 2022].

[33] R. Aroca, "Análise de sistemas operacionais de tempo real para aplicações de robótica e automação," 2008.

[34] SQLite, "SQLite," [Online]. Available: https://www.sqlite.org/about.html. [Acedido em 27 01 2022].

[35] "O que são máquinas virtuais (VM)?," 2023. [Online]. Available: https://azure.microsoft.com/pt-pt/resources/cloud-computing-dictionary/what-is-a-virtual-machine. [Acedido em 2023 01 21].

[36] "Virtualbox," [Online]. Available: https://www.virtualbox.org/manual/ch01.html#virt-why-useful. [Acedido em 09 09 2021].

[37] "DB Browser for SQLite," [Online]. Available: https://sqlitebrowser.org/. [Acedido em 21 09 2021].

[38] R. Chandel, "Comprehensive Guide on FTK Imager," 6 11 2020. [Online]. Available: https://www.hackingarticles.in/comprehensive-guide-on-ftk-imager/. [Acedido em 05 05 2021].

[39] "Autopsy," 2022. [Online]. Available: https://www.autopsy.com/. [Acedido em 2022 10 8].

[40] "Open Source Digital Forensics," [Online]. Available: https://www.sleuthkit.org/. [Acedido em 08 08 2020].

[41] Basis Technology., "Autopsy Workflow," 19 10 2015. [Online]. Available: https://sleuthkit.org/autopsy/docs/user-docs/3.1/workflow_page.html. [Acedido em 10 10 2020].

[42] E. Bates, "Digital vehicle forensics data recivery from BMW infotainment systems," *The APB a leader in vehicle crime training and information,* vol. 26, nº 2, 2020.

[43] A. Saxena, "Infochips," 03 02 2020. [Online]. Available: https://www.einfochips.com/blog/everything-you-need-to-know-about-in-vehicle-infotainment-system/. [Acedido em 09 09 2021].

[44] 20 01 2014. [Online]. Available: https://f30.bimmerpost.com/forums/showthread.php?t=921688&page=14. [Acedido em 21 02 2022].

[45] Bayerische Motoren Werke, BMW Nav Pro User Manual - Navigation, entertainment, communication, Munich, 2015.

[46] INTERPOL, "Guidelines for digital forensics first responses, Best practices for search and seizure," pp. 49-52, 03 2021.

[47] E. Bates, "Evidence Presevation from infotainment systems".

[48] M. Evans, "Car Forensics - A Starting Point," 30 06 2017. [Online]. Available: https://nop.ninja/b254dd7da1f1463da566b48b09c43aab.php. [Acedido em 23 06 2022].

[49] "O que é o código IMEI e como você pode usá-lo para bloquear e desbloquear seu celular," 29 01 2018. [Online]. Available: https://www.bbc.com/portuguese/brasil-42775211.

[50] M. Frade, "Mobile Devices Forensics," 2016.

[51] A. Longmuir, "What Is an International Mobile Subscriber Identity (IMSI)," 22 12 2020. [Online]. Available: https://www.emnify.com/iot-glossary/imsi. [Acedido em 01 01 2021].

[52] E. ADAMS, "The Best Automotive Infotainment Systems You Can Buy," 17 08 2020. [Online]. Available: https://www.gearpatrol.com/cars/a447579/the-7-best-car-infotainment-systems-you-can-buy/. [Acedido em 2022 02 21].

[53] "The iVe Ecosystem," Berla, 2022. [Online]. Available: https://berla.co/ecosystem/. [Acedido em 02 01 2022].

[54] A. Klavmark e T. Vikingsson, "Study on open source in-vehicle infotainment (IVI) software platforms," 2015.

[55] P. Silva, "Captura e análise de conteúdos de memória em sistemas computacionais ligados".

[56] B. QNX, "Partitions," 14 01 2021. [Online]. Available: http://www.qnx.com/developers/docs/7.0.0/index.html#com.qnx.doc.neutrino.sys_arch/topic/fsys_Partitions.html. [Acedido em 05 05 2021].

[57] B. QNX, “OS partitions,” 14 01 2021. [Online]. Available: http://www.qnx.com/developers/docs/qnx_4.25_docs/qnx4/sysarch/fsys.html#RAW VOLUMES. [Acedido em 02 02 2021].

[58] SQLite, “Arquitetura do SQLite,” SQLite, [Online]. Available: https://www.sqlite.org/arch.html. [Acedido em 06 08 2022].

[59] J. Knispel, “Automotive Forensik - Forensische Analyse gangiger Car Infotainment Systeme”.

[60] “What is Kali Linux?,” 09 09 2022. [Online]. Available: https://www.kali.org/docs/introduction/what-is-kali-linux/. [Acedido em 07 07 2021].

[61] “Pycharm Tutorial,” 2022. [Online]. Available: https://www.tutorialspoint.com/pycharm/index.htm. [Acedido em 12 12 2021].

Anexo A - Código ingest module sistema IVI NBT

```
# Sample module in the public domain. Feel free to use this as a template
# This software is free, you can use and modify
#
# Create ingest module by Ricardo Marques Master's student in
Cybersecurity and Forensic Informatics
# Polytechnic Institute of Leiria - Portugal
#
# This ingest module allows you to investigate SQLite database of
# BMW brand In-Vehicle Infotainment systems, NBT model year 2017
# Find data such as contacts, messages, Bluetooth mac address
# MEI, IMSI, call history, which devices were connected
# music tastes, web history, cookies and musics groups
#
# This ingest module was used Autopsy version 4.18.0
#
# Instructions for using this ingest module:
# Tools - Python Plugins and put ingest module in that folder



import jarray
import inspect
import os
from java.lang import Class
from java.lang import System
from java.sql  import DriverManager, SQLException
from java.util.logging import Level
from java.util import ArrayList
from java.io import File
from org.sleuthkit.datamodel import SleuthkitCase
from org.sleuthkit.datamodel import AbstractFile
from org.sleuthkit.datamodel import ReadContentInputStream
from org.sleuthkit.datamodel import BlackboardArtifact
from org.sleuthkit.datamodel import BlackboardAttribute
from org.sleuthkit.autopsy.ingest import IngestModule
from org.sleuthkit.autopsy.ingest.IngestModule import
IngestModuleException
from org.sleuthkit.autopsy.ingest import DataSourceIngestModule
from org.sleuthkit.autopsy.ingest import IngestModuleFactoryAdapter
from org.sleuthkit.autopsy.ingest import IngestMessage
from org.sleuthkit.autopsy.ingest import IngestServices
from org.sleuthkit.autopsy.ingest import ModuleDataEvent
from org.sleuthkit.autopsy.coreutils import Logger
from org.sleuthkit.autopsy.casemodule import Case
from org.sleuthkit.autopsy.datamodel import ContentUtils
from org.sleuthkit.autopsy.casemodule.services import Services
from org.sleuthkit.autopsy.casemodule.services import FileManager
from org.sleuthkit.autopsy.casemodule.services import Blackboard


class IviBmwDbIngestModuleFactory(IngestModuleFactoryAdapter):

    moduleName = "Infotainment BMW NBT"

    def getModuleDisplayName(self):
```

```
        return self.moduleName


    def getModuleDescription(self):
        return "extract phone numbers,email,address,country,call and
message,connected devices," \
               "bluetooth,macaddress,EMEI,IMSI,model smartphone,favorite
songs"

    def getModuleVersionNumber(self):
        return "1.0"

    def isDataSourceIngestModuleFactory(self):
        return True

    def createDataSourceIngestModule(self, ingestOptions):
        # TODO: Change the class name to the name you'll make below
        return IviBmwDbIngestModule()

class IviBmwDbIngestModule(DataSourceIngestModule):

    _logger = Logger.getLogger(IviBmwDbIngestModuleFactory.moduleName)

    def log(self, level, msg):
        self._logger.logp(level, self.__class__.__name__,
inspect.stack()[1][3], msg)

    def __init__(self):
        self.context = None



    def startUp(self, context):
            self.context = context


    def process(self, dataSource, progressBar):

        # we don't know how much work there is yet
        progressBar.switchToIndeterminate()

        # Use blackboard class to index blackboard artifacts for keyword
search
        blackboard = Case.getCurrentCase().getServices().getBlackboard()

        # Find files named contacts.db, regardless of parent path
        fileManager =
Case.getCurrentCase().getServices().getFileManager()
        files = fileManager.findFiles(dataSource, "contactbook_%.db")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1
```

```
            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(),
                                        str(file.getId()) + ".db")
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) "
                         + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the contacts table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT
contact_card_phone.Contact_ID, "
                        "contact_card_phone.GivenName,
contact_card_phone.FamilyName, "
                        "contact_card_phone.Url,
contact_card_phone.organisation FROM "
                        "contact_card_phone "
                        "ORDER BY contact_card_phone.GivenName")

            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table "
                                     "(" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    Contact_ID = resultSet.getString("Contact_ID")
                    GivenName = resultSet.getString("GivenName")
                    FamilyName = resultSet.getString("FamilyName")
                    Url = resultSet.getString("Url")
                    organisation = resultSet.getString("organisation")


                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table"
                                         " (" + e.getMessage() + ")")


                # Make an artifact on the blackboard, TSK_CONTACT and
give it attributes for each of the fields
                art =
file.newArtifact(BlackboardArtifact.ARTIFACT_TYPE.TSK_CONTACT)
                family_name_att_type = blackboard.getOrAddAttributeType\

('BMW_FAMILY_NAME_TYPE',BlackboardAttribute.TSK_BLACKBOARD_ATTRIBUTE_VALU
```

```
E_TYPE.STRING, "FamilyName")
                attributes = ArrayList()


attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_USER_ID.getTypeID(),

IviBmwDbIngestModuleFactory.moduleName, Contact_ID))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_NAME.getTypeID(),

IviBmwDbIngestModuleFactory.moduleName, GivenName))

attributes.add(BlackboardAttribute(family_name_att_type,IviBmwDbIngestMod
uleFactory.moduleName, FamilyName))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_URL.getTypeID(),

IviBmwDbIngestModuleFactory.moduleName, Url))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_ORGANIZATION.getTypeID(),

IviBmwDbIngestModuleFactory.moduleName, organisation))


                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(
                ModuleDataEvent(IviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_CONTACT, None))


        #contact phone
        artttId = blackboard.getOrAddArtifactType("TSK_CONTACT_PHONE",
"Contact Phone")
        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
```

```
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db")
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the contacts table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT
contact_card_phone.Contact_ID, contact_card_phone.GivenName, "
                            "contact_card_phone.FamilyName,
contact_card_phone.AdditionalName, "
                            "contact_card_phone.Url,
contact_card_phone.organisation, "
                            "phone_data_phone.PhoneNumber FROM
contact_card_phone "
                            "JOIN phone_data_phone ON
contact_card_phone.Contact_ID = phone_data_phone.Contact_ID "
                                    "ORDER BY
contact_card_phone.GivenName")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    Contact_ID = resultSet.getString("Contact_ID")
                    GivenName = resultSet.getString("GivenName")
                    FamilyName = resultSet.getString("FamilyName")
                    PhoneNumber = resultSet.getString("PhoneNumber")


                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")


                # Make an artifact on the blackboard, TSK_CONTACT and
give it attributes for each of the fields
                art = file.newArtifact(arttttId.getTypeID())
                family_name_att_type =
blackboard.getOrAddAttributeType('BMW_FAMILY_NAME_TYPE',BlackboardAttribu
te.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "FamilyName")
                attributes = ArrayList()


attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_USER_ID.getTypeID(), IviBmwDbIngestModuleFactory.moduleName,
Contact_ID))
```

```
attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_NAME.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, GivenName))

attributes.add(BlackboardAttribute(family_name_att_type,IviBmwDbIngestMod
uleFactory.moduleName, FamilyName))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_PHONE_NUMBER.getTypeID(), IviBmwDbIngestModuleFactory.moduleName,
PhoneNumber))


                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(
                ModuleDataEvent(IviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_CONTACT, None))


        #contact email
        artttttIId = blackboard.getOrAddArtifactType("TSK_CONTACT_EMAIL",
"Contact Email")
        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db")
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the contacts table in the database and get all
columns.
```

```
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT
contact_card_phone.Contact_ID, contact_card_phone.GivenName, "
                            "contact_card_phone.FamilyName,
msg_data_phone.EmailAddr FROM contact_card_phone "
                                "JOIN msg_data_phone ON
contact_card_phone.Contact_ID = msg_data_phone.Contact_ID "
                            "ORDER BY contact_card_phone.GivenName")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    Contact_ID = resultSet.getString("Contact_ID")
                    GivenName = resultSet.getString("GivenName")
                    FamilyName = resultSet.getString("FamilyName")
                    EmailAddr = resultSet.getString("EmailAddr")


                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")


                # Make an artifact on the blackboard, TSK_CONTACT and
give it attributes for each of the fields
                art = file.newArtifact(artttttIId.getTypeID())
                family_name_att_type =
blackboard.getOrAddAttributeType('BMW_FAMILY_NAME_TYPE',BlackboardAttribu
te.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "FamilyName")
                attributes = ArrayList()


attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_USER_ID.getTypeID(), IviBmwDbIngestModuleFactory.moduleName,
Contact_ID))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_NAME.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, GivenName))

attributes.add(BlackboardAttribute(family_name_att_type,IviBmwDbIngestMod
uleFactory.moduleName, FamilyName))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_EMAIL.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, EmailAddr))



                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())
```

```
            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(
                ModuleDataEvent(IviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_CONTACT, None))

        #contact address
        artttt IIId =
blackboard.getOrAddArtifactType("TSK_CONTACT_ADDRESS", "Contact Address")
        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db")
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the contacts table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT
contact_card_phone.Contact_ID, contact_card_phone.GivenName, "
                            "contact_card_phone.FamilyName,
contact_card_phone.AdditionalName, "
                            "contact_card_phone.Url,
contact_card_phone.organisation, "
                            "address_phone.StreetHousenumber,
address_phone.City, "
                            "address_phone.Country,
address_phone.Postalcode FROM contact_card_phone "
                            "JOIN address_phone ON
contact_card_phone.Contact_ID = address_phone.Contact_ID "
                                    "WHERE address_phone.crosssum > 0 "
                            "ORDER BY contact_card_phone.GivenName")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK
```

```
            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    Contact_ID = resultSet.getString("Contact_ID")
                    GivenName = resultSet.getString("GivenName")
                    FamilyName = resultSet.getString("FamilyName")
                    StreetHousenumber =
resultSet.getString("StreetHousenumber")
                    City = resultSet.getString("City")
                    Country = resultSet.getString("Country")
                    Postalcode = resultSet.getString("Postalcode")

                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")


                # Make an artifact on the blackboard, TSK_CONTACT and
give it attributes for each of the fields
                art = file.newArtifact(arttttIIId.getTypeID())
                postal_code_att_type =
blackboard.getOrAddAttributeType('BMW_POSTAL_CODE_TYPE',BlackboardAttribu
te.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "PostalCode")
                family_name_att_type =
blackboard.getOrAddAttributeType('BMW_FAMILY_NAME_TYPE',BlackboardAttribu
te.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "FamilyName")

                attributes = ArrayList()


attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_USER_ID.getTypeID(), IviBmwDbIngestModuleFactory.moduleName,
Contact_ID))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_NAME.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, GivenName))

attributes.add(BlackboardAttribute(family_name_att_type,IviBmwDbIngestMod
uleFactory.moduleName, FamilyName))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_LOCATION.getTypeID(), IviBmwDbIngestModuleFactory.moduleName,
StreetHousenumber))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_CITY.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, City))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_COUNTRY.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, Country))

attributes.add(BlackboardAttribute(postal_code_att_type,IviBmwDbIngestMod
uleFactory.moduleName, Postalcode))



                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
```

```
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(
                ModuleDataEvent(IviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_CONTACT, None))




        #bluetooth
        arttId = blackboard.getOrAddArtifactType("TSK_BLUETOOTH_ADDRESS",
"Bluetooth Address")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db")
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the bluetooth table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT Origin, BtAddress
FROM bluetooth")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    Origin = resultSet.getString("Origin")
```

```
                    BtAddress = resultSet.getString("BtAddress")


                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")


                # Make an artifact on the blackboard,
TSK_BLUETOOTH_PAIRING and give it attributes for each of the fields
                art = file.newArtifact(arttId.getTypeID())

                bluetooth_address_att_type =
blackboard.getOrAddAttributeType('BMW_BLUETOOTH_ADDRESS_TYPE',BlackboardA
ttribute.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "BtAddress")

                attributes = ArrayList()

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_ID.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, Origin))

attributes.add(BlackboardAttribute(bluetooth_address_att_type,IviBmwDbIng
estModuleFactory.moduleName, BtAddress))

                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(
                ModuleDataEvent(IviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_METADATA, None))




        #callstacks
        files = fileManager.findFiles(dataSource, "pm800%.a")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
```

```
+ ".db" )
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the callstacks table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT CALLSTACKS.ID,
CALLSTACKS.FN, CALLSTACKS.TEL_NR, STRFTIME('%s', CALLSTACKS.TIMESTAMP) AS
TIMESTAMP FROM CALLSTACKS")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    ID = resultSet.getString("ID")
                    if resultSet.getString("FN") == None:
                        FN = ""
                    else:
                        FN = resultSet.getString("FN")
                    TEL_NR = resultSet.getString("TEL_NR")
                    TIMESTAMP = resultSet.getInt("TIMESTAMP")


                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")

                # Make an artifact on the blackboard, TSK_CALLLOG and
give it attributes for each of the fields
                art =
file.newArtifact(BlackboardArtifact.ARTIFACT_TYPE.TSK_CALLLOG)
                attributes = ArrayList()

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_ID.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, ID))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_NAME.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, FN))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_PHONE_NUMBER.getTypeID(), IviBmwDbIngestModuleFactory.moduleName,
TEL_NR))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_DATETIME.getTypeID(), IviBmwDbIngestModuleFactory.moduleName,
long(TIMESTAMP)))
```

```
                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(
                ModuleDataEvent(IviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_CALLLOG, None))




        #bluetooth pairing, BluetoothAddress, EMEI, IMSI, MODEL TELEPHONE
        files = fileManager.findFiles(dataSource, "p%.db")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db")
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the bluetooth pairing, BluetoothAddress, EMEI, IMSI,
MODEL TELEPHONE table in the database and get all columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT SID, INFO_KEY,
INFO_VALUE FROM CE_DEVICE_INFO WHERE INFO_KEY = 'IMEI' OR INFO_KEY =
'IMSI' OR INFO_KEY = 'BluetoothAddress' or INFO_KEY = 'Model' ORDER BY
SID")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
```

```
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    SID = resultSet.getString("SID")
                    INFO_KEY = resultSet.getString("INFO_KEY")
                    INFO_VALUE = resultSet.getString("INFO_VALUE")


                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")


                # Make an artifact on the blackboard,
TSK_BLUETOOTH_PAIRING and give it attributes for each of the fields
                art =
file.newArtifact(BlackboardArtifact.ARTIFACT_TYPE.TSK_BLUETOOTH_PAIRING)
                attributes = ArrayList()

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_ID.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, SID))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_DESCRIPTION.getTypeID(), IviBmwDbIngestModuleFactory.moduleName,
INFO_KEY))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_DEVICE_ID.getTypeID(), IviBmwDbIngestModuleFactory.moduleName,
INFO_VALUE))

                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(
                ModuleDataEvent(IviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_BLUETOOTH_PAIRING,
None))



        #BROWSER
        files = fileManager.findFiles(dataSource, "BrowserUrls.db")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK
```

```
            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db")
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the BROWSER table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT urls.id,
urls.title, urls.url, visits.datevisit FROM urls LEFT JOIN visits")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    id = resultSet.getString("id")
                    title = resultSet.getString("title")
                    url = resultSet.getString("url")
                    datevisit = resultSet.getInt("datevisit")
                    urlid = resultSet.getString("urlid")



                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")


                # Make an artifact on the blackboard, TSK_WEB_HISTORY and
give it attributes for each of the fields
                art =
file.newArtifact(BlackboardArtifact.ARTIFACT_TYPE.TSK_WEB_HISTORY)
                attributes = ArrayList()

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_ID.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, id))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_TITLE.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, title))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_URL.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, url))
```

```
attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_DATETIME_ACCESSED.getTypeID(), IviBmwDbIngestModuleFactory.moduleName,
datevisit))

                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(
                ModuleDataEvent(IviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_WEB_HISTORY, None))


        #COOKIES
        files = fileManager.findFiles(dataSource, "cookie.db")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db")
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the COOKIES table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT cookies.name,
cookies.host, cookies.path, cookies.lastAccessed FROM cookies")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
```

```
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    name = resultSet.getString("name")
                    host = resultSet.getString("host")
                    path = resultSet.getString("path")
                    lastAccessed = resultSet.getInt("lastAccessed")




                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")


                # Make an artifact on the blackboard, TSK_WEB_COOKIE and
give it attributes for each of the fields
                art =
file.newArtifact(BlackboardArtifact.ARTIFACT_TYPE.TSK_WEB_COOKIE)
                attributes = ArrayList()

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_NAME.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, name))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_URL.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, host))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_PATH.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, path))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_DATETIME_ACCESSED.getTypeID(), IviBmwDbIngestModuleFactory.moduleName,
lastAccessed))

                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(
                ModuleDataEvent(IviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_WEB_COOKIE, None))



        #messages
        files = fileManager.findFiles(dataSource, "f2%.sqlite")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:
```

```
            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db" )
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the messages table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery('SELECT messages.id,
messages.fromPhoneNumber, strftime("%s", substr(date,1,4) || "-" ||
substr(date,5,2) || "-" || substr(date,7,2) || "T" || substr(date,9,2) ||
":" || substr(date,11,2) || ":" || substr(date,13,2)) as newdate,
messages.subject FROM messages')
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    id = resultSet.getString("id")
                    fromPhoneNumber =
resultSet.getString("fromPhoneNumber")
                    date = resultSet.getLong("newdate")
                    subject = resultSet.getString("subject")

                    art =
file.newArtifact(BlackboardArtifact.ARTIFACT_TYPE.TSK_MESSAGE)

                    from_number_att_type =
blackboard.getOrAddAttributeType('BMW_FROM_NUMBER_TYPE',BlackboardAttribu
te.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "fromPhoneNumber")

                    attributes = ArrayList()


attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_ID.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, id))

attributes.add(BlackboardAttribute(from_number_att_type,IviBmwDbIngestMod
```

```
uleFactory.moduleName, fromPhoneNumber))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_DATETIME.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, date))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_TEXT.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, subject))



                    art.addAttributes(attributes)
                    try:

Case.getCurrentCase().getSleuthkitCase().getBlackboard().postArtifact(art
, IviBmwDbIngestModuleFactory.moduleName)
                    except Blackboard.BlackboardException as e:
                        self.log(Level.SEVERE, "Error posting artifact "
+ art.getDisplayName())

                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")



        #mme_mediastores
        files = fileManager.findFiles(dataSource, "mme%")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db" )
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the mme_mediastores table in the database and get all
columns.
```

```
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT msid, lastseen,
mssname, name, identifier, mountpath FROM mediastores")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    msid = resultSet.getString("msid")
                    lastseen = resultSet.getLong("lastseen")
                    mssname = resultSet.getString("mssname")
                    name = resultSet.getString("name")
                    identifier = resultSet.getString("identifier")
                    mountpath = resultSet.getString("mountpath")


                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")

                # Make an artifact on the blackboard, TSK_DEVICE_INFO and
give it attributes for each of the fields
                art =
file.newArtifact(BlackboardArtifact.ARTIFACT_TYPE.TSK_DEVICE_INFO)
                attributes = ArrayList()

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_ID.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, msid))

                timevalue = lastseen/1000000000

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_DATETIME.getTypeID(), IviBmwDbIngestModuleFactory.moduleName,
timevalue))


attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_DESCRIPTION.getTypeID(), IviBmwDbIngestModuleFactory.moduleName,
mssname))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_NAME.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, name))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_DEVICE_ID.getTypeID(), IviBmwDbIngestModuleFactory.moduleName,
identifier))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_PATH.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, mountpath))


                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
```

```
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(
                ModuleDataEvent(IviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_DEVICE_INFO, None))




        #music and groups
        artId = blackboard.getOrAddArtifactType("TSK_MUSIC_GROUPS",
"Music Groups")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db" )
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the music and groups  table in the database and get
all columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT
categorydata_custom.name FROM categorydata_custom")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    name = resultSet.getString("name")
```

```
                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")

                # Make an artifact on the blackboard, TSK_DOWNLOAD_SOURCE
and give it attributes for each of the fields
                art = file.newArtifact(artId.getTypeID())
                attributes = ArrayList()


attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_NAME.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, name))



                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(
                ModuleDataEvent(IviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_METADATA, None))


        #software
        artIId = blackboard.getOrAddArtifactType("TSK_SOFTWARE_INFO",
"Software info")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db" )
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
```

```
                return IngestModule.ProcessResult.OK

            # Query the software table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT
software_info.version FROM software_info")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    version = resultSet.getString("version")


                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")

                # Make an artifact on the blackboard, TSK_DOWNLOAD_SOURCE
and give it attributes for each of the fields
                art = file.newArtifact(artIId.getTypeID())
                attributes = ArrayList()


attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_VERSION.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, version))



                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(
                ModuleDataEvent(IviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_METADATA, None))


        #usbdetails
        artIIId =
blackboard.getOrAddArtifactType("TSK_USB_DEVICEDETAILS", "Usb device
details")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
```

```
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db" )
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the usbdetails table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT deviceserialno,
lastseen FROM usbdevicedetails")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    deviceserialno =
resultSet.getString("deviceserialno")
                    lastseen = resultSet.getLong("lastseen")

                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")

                # Make an artifact on the blackboard, TSK_DOWNLOAD_SOURCE
and give it attributes for each of the fields
                art = file.newArtifact(artIIId.getTypeID())

                device_name_att_type =
blackboard.getOrAddAttributeType('BMW_DEVICE_NAME_TYPE',BlackboardAttribu
te.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "deviceserialno")

                attributes = ArrayList()



attributes.add(BlackboardAttribute(device_name_att_type,IviBmwDbIngestMod
uleFactory.moduleName, deviceserialno))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
```

```
_DATETIME_ACCESSED.getTypeID(), IviBmwDbIngestModuleFactory.moduleName,
lastseen))


                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(
                ModuleDataEvent(IviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_METADATA, None))


        #folders
        artIIIId = blackboard.getOrAddArtifactType("TSK_FOLDERS",
"Folders")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db" )
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the folders table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT foldername,
last_sync, basepath FROM folders")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
```

```
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    foldername = resultSet.getString("foldername")
                    last_sync = resultSet.getLong("last_sync")
                    basepath = resultSet.getString("basepath")


                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")

                # Make an artifact on the blackboard, TSK_DOWNLOAD_SOURCE
and give it attributes for each of the fields
                art = file.newArtifact(artIIIId.getTypeID())

                folder_name_att_type =
blackboard.getOrAddAttributeType('BMW_FOLDER_NAME_TYPE',BlackboardAttribu
te.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "foldername")

                attributes = ArrayList()


attributes.add(BlackboardAttribute(folder_name_att_type,IviBmwDbIngestMod
uleFactory.moduleName, foldername))
                timevalue = last_sync/1000000000

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_DATETIME_MODIFIED.getTypeID(), IviBmwDbIngestModuleFactory.moduleName,
timevalue))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_PATH.getTypeID(), IviBmwDbIngestModuleFactory.moduleName, basepath))

                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(
                ModuleDataEvent(IviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_METADATA, None))




        #Library albuns
        artIIIIId = blackboard.getOrAddArtifactType("TSK_LIBRARY_ALBUNS",
"Library albums")

        numFiles = len(files)
```

```
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db" )
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the Library albuns table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT
library_albums.album FROM library_albums")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    album = resultSet.getString("album")


                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")

                # Make an artifact on the blackboard, TSK_DOWNLOAD_SOURCE
and give it attributes for each of the fields
                art = file.newArtifact(artIIIIId.getTypeID())
                library_albums_att_type =
blackboard.getOrAddAttributeType('BMW_LIBRARY_ALBUMS_TYPE',BlackboardAttr
ibute.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "album")
                attributes = ArrayList()


attributes.add(BlackboardAttribute(library_albums_att_type,IviBmwDbIngest
ModuleFactory.moduleName, album))
```

```
                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(
                ModuleDataEvent(IviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_METADATA, None))


        #Library artists
        artIIIIIId =
blackboard.getOrAddArtifactType("TSK_LIBRARY_ARTISTS", "Library artists")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db" )
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the Library artists  table in the database and get
all columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT
library_artists.artist FROM library_artists")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
```

```
                try:
                    artist = resultSet.getString("artist")


                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")

                # Make an artifact on the blackboard, TSK_DOWNLOAD_SOURCE
and give it attributes for each of the fields
                art = file.newArtifact(artIIIIIId.getTypeID())
                library_artists_att_type =
blackboard.getOrAddAttributeType('BMW_LIBRARY_ARTISTS_TYPE',BlackboardAtt
ribute.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "artist")
                attributes = ArrayList()


attributes.add(BlackboardAttribute(library_artists_att_type,IviBmwDbInges
tModuleFactory.moduleName, artist))

                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(
                ModuleDataEvent(IviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_METADATA, None))




        #Post a message to the ingest messages in box.
        message =
IngestMessage.createMessage(IngestMessage.MessageType.DATA,
            "Sample Jython Data Source Ingest Module", "Found %d files" %
fileCount)
        IngestServices.getInstance().postMessage(message)

        return IngestModule.ProcessResult.OK
```

Anexo B - Código ingest module sistema IVI CIC

```
# Sample module in the public domain. Feel free to use this as a template
# This software is free, you can use and modify
#
# Create ingest module by Ricardo Marques Master's student in
Cybersecurity and Forensic Informatics
# Polytechnic Institute of Leiria - Portugal
#
# This ingest module allows you to investigate SQLite database of
# BMW brand In-Vehicle Infotainment systems CIC model year 2010 and 2012
# Find data such as contacts
# Bluetooth mac address
# which devices were connected
#
#
# This ingest module was used Autopsy version 4.18.0
#
# Instructions for using this ingest module:
# Tools - Python Plugins and put ingest module in that folder
#
#


import jarray
import inspect
import os
from java.lang import Class
from java.lang import System
from java.sql  import DriverManager, SQLException
from java.util.logging import Level
from java.util import ArrayList
from java.io import File
from org.sleuthkit.datamodel import SleuthkitCase
from org.sleuthkit.datamodel import AbstractFile
from org.sleuthkit.datamodel import ReadContentInputStream
from org.sleuthkit.datamodel import BlackboardArtifact
from org.sleuthkit.datamodel import BlackboardAttribute
from org.sleuthkit.autopsy.ingest import IngestModule
from org.sleuthkit.autopsy.ingest.IngestModule import
IngestModuleException
from org.sleuthkit.autopsy.ingest import DataSourceIngestModule
from org.sleuthkit.autopsy.ingest import IngestModuleFactoryAdapter
from org.sleuthkit.autopsy.ingest import IngestMessage
from org.sleuthkit.autopsy.ingest import IngestServices
from org.sleuthkit.autopsy.ingest import ModuleDataEvent
from org.sleuthkit.autopsy.coreutils import Logger
from org.sleuthkit.autopsy.casemodule import Case
from org.sleuthkit.autopsy.datamodel import ContentUtils
from org.sleuthkit.autopsy.casemodule.services import Services
from org.sleuthkit.autopsy.casemodule.services import FileManager
from org.sleuthkit.autopsy.casemodule.services import Blackboard


# Factory that defines the name and details of the module and allows
Autopsy
# to create instances of the modules that will do the analysis.
# Rename this to something more specific. Search and replace for it
```

```
because it is used a few times
class CicIviBmwDbIngestModuleFactory(IngestModuleFactoryAdapter):

    # TODO: give it a unique name.
    moduleName = "Infotainment BMW CIC"

    def getModuleDisplayName(self):
        return self.moduleName

    # TODO: Give it a description
    def getModuleDescription(self):
        return "extract phone
numbers,email,address,country,bluetooth,macaddress,devices were
connected"

    def getModuleVersionNumber(self):
        return "1.0"

    def isDataSourceIngestModuleFactory(self):
        return True

    def createDataSourceIngestModule(self, ingestOptions):
        #Change the class name to the name you'll make below
        return CicIviBmwDbIngestModule()


#Data Source-level ingest module.  One gets created per data source.
#Rename this to something more specific.
class CicIviBmwDbIngestModule(DataSourceIngestModule):

    _logger = Logger.getLogger(CicIviBmwDbIngestModuleFactory.moduleName)

    def log(self, level, msg):
        self._logger.logp(level, self.__class__.__name__,
inspect.stack()[1][3], msg)

    def __init__(self):
        self.context = None

    # Where any setup and configuration is done
    # 'context' is an instance of
org.sleuthkit.autopsy.ingest.IngestJobContext.
    # See: http://sleuthkit.org/autopsy/docs/api-
docs/4.4/classorg_1_1sleuthkit_1_1autopsy_1_1ingest_1_1_ingest_job_contex
t.html
    # Add any setup code that you need here.
    def startUp(self, context):

        # Throw an IngestModule.IngestModuleException exception if there
was a problem setting up

        self.context = context

    # Where the analysis is done.
    # The 'dataSource' object being passed in is of type
org.sleuthkit.datamodel.Content.
    # See: http://www.sleuthkit.org/sleuthkit/docs/jni-
docs/4.4/interfaceorg_1_1sleuthkit_1_1datamodel_1_1_content.html
    # 'progressBar' is of type
org.sleuthkit.autopsy.ingest.DataSourceIngestModuleProgress
    # See: http://sleuthkit.org/autopsy/docs/api-
```

```
docs/4.4/classorg_1_1sleuthkit_1_1autopsy_1_1ingest_1_1_data_source_inges
t_module_progress.html
    # Add your analysis code in here.
    def process(self, dataSource, progressBar):

        # we don't know how much work there is yet
        progressBar.switchToIndeterminate()

        # Use blackboard class to index blackboard artifacts for keyword
search
        blackboard = Case.getCurrentCase().getServices().getBlackboard()

        # Find files named contactbook, regardless of parent path
        fileManager =
Case.getCurrentCase().getServices().getFileManager()
        files = fileManager.findFiles(dataSource, "contactbook_%.db")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db")
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the contacts table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT
contact_card_phone.Contact_ID, "
                                              "contact_card_phone.GivenName,
contact_card_phone.FamilyName, "
                                              "contact_card_phone.organisation FROM
contact_card_phone "
                                              "ORDER BY
contact_card_phone.GivenName")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
```

```
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    Contact_ID = resultSet.getString("Contact_ID")
                    GivenName = resultSet.getString("GivenName")
                    FamilyName = resultSet.getString("FamilyName")
                    organisation = resultSet.getString("organisation")


                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")


                # Make an artifact on the blackboard, TSK_CONTACT and
give it attributes for each of the fields
                art =
file.newArtifact(BlackboardArtifact.ARTIFACT_TYPE.TSK_CONTACT)
                family_name_att_type =
blackboard.getOrAddAttributeType('BMW_FAMILY_NAME_TYPE',BlackboardAttribu
te.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "FamilyName")
                attributes = ArrayList()


attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_USER_ID.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName,
Contact_ID))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_NAME.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName, GivenName))

attributes.add(BlackboardAttribute(family_name_att_type,CicIviBmwDbIngest
ModuleFactory.moduleName, FamilyName))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_ORGANIZATION.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName,
organisation))




                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(

ModuleDataEvent(CicIviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_CONTACT, None))
```

```
        #database conctactbook search contact phone

        files = fileManager.findFiles(dataSource, "contactbook_%.db")

        artttId = blackboard.getOrAddArtifactType("TSK_CONTACT_PHONE",
"Contact Phone")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db")
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the contact phone table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT
contact_card_phone.Contact_ID, contact_card_phone.GivenName, "
                            "contact_card_phone.FamilyName,"
                            "phone_data_phone.PhoneNumber FROM
contact_card_phone "
                            "JOIN phone_data_phone ON
contact_card_phone.Contact_ID = phone_data_phone.Contact_ID "
                                    "ORDER BY
contact_card_phone.GivenName")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
```

```
                    Contact_ID = resultSet.getString("Contact_ID")
                    GivenName = resultSet.getString("GivenName")
                    FamilyName = resultSet.getString("FamilyName")
                    PhoneNumber = resultSet.getString("PhoneNumber")


                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")


                # Make an artifact on the blackboard, TSK_CONTACT and
give it attributes for each of the fields
                art = file.newArtifact(artttId.getTypeID())
                family_name_att_type =
blackboard.getOrAddAttributeType('BMW_FAMILY_NAME_TYPE',BlackboardAttribu
te.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "FamilyName")
                attributes = ArrayList()


attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_USER_ID.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName,
Contact_ID))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_NAME.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName, GivenName))

attributes.add(BlackboardAttribute(family_name_att_type,CicIviBmwDbIngest
ModuleFactory.moduleName, FamilyName))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_PHONE_NUMBER.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName,
PhoneNumber))


                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(

ModuleDataEvent(CicIviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_CONTACT, None))



        #database contactbook search contact email
        files = fileManager.findFiles(dataSource, "contactbook_%.db")

        artttIId = blackboard.getOrAddArtifactType("TSK_CONTACT_EMAIL",
"Contact Email")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
```

```
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db")
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the contact email table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT
contact_card_phone.Contact_ID, contact_card_phone.GivenName, "
                            "contact_card_phone.FamilyName,
msg_data_phone.EmailAddr FROM contact_card_phone "
                                "JOIN msg_data_phone ON
contact_card_phone.Contact_ID = msg_data_phone.Contact_ID "
                            "ORDER BY contact_card_phone.GivenName")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    Contact_ID = resultSet.getString("Contact_ID")
                    GivenName = resultSet.getString("GivenName")
                    FamilyName = resultSet.getString("FamilyName")
                    EmailAddr = resultSet.getString("EmailAddr")


                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")


                # Make an artifact on the blackboard, TSK_CONTACT_EMAIL
and give it attributes for each of the fields
                art = file.newArtifact(arttttIId.getTypeID())
                family_name_att_type =
```

```
blackboard.getOrAddAttributeType('BMW_FAMILY_NAME_TYPE',BlackboardAttribu
te.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "FamilyName")
                attributes = ArrayList()


attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_USER_ID.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName,
Contact_ID))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_NAME.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName, GivenName))

attributes.add(BlackboardAttribute(family_name_att_type,CicIviBmwDbIngest
ModuleFactory.moduleName, FamilyName))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_EMAIL.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName,
EmailAddr))



                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(

ModuleDataEvent(CicIviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_CONTACT, None))



        #database contactbook search contact address
        files = fileManager.findFiles(dataSource, "contactbook_%.db")

        arttttIIId =
blackboard.getOrAddArtifactType("TSK_CONTACT_ADDRESS", "Contact Address")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
```

```
+ ".db")
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the contact address table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT
contact_card_phone.Contact_ID, contact_card_phone.GivenName, "
                            "contact_card_phone.FamilyName, "
                            "address_phone.StreetHousenumber,
address_phone.City, "
                            "address_phone.Country,
address_phone.Postalcode FROM contact_card_phone "
                            "JOIN address_phone ON
contact_card_phone.Contact_ID = address_phone.Contact_ID "
                                    "WHERE address_phone.crosssum > 0 "
                            "ORDER BY contact_card_phone.GivenName")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    Contact_ID = resultSet.getString("Contact_ID")
                    GivenName = resultSet.getString("GivenName")
                    FamilyName = resultSet.getString("FamilyName")
                    StreetHousenumber =
resultSet.getString("StreetHousenumber")
                    City = resultSet.getString("City")
                    Country = resultSet.getString("Country")
                    Postalcode = resultSet.getString("Postalcode")

                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")


                # Make an artifact on the blackboard, TSK_CONTACT_ADDRESS
and give it attributes for each of the fields
                art = file.newArtifact(artttttIIId.getTypeID())
                postal_code_att_type =
blackboard.getOrAddAttributeType('BMW_POSTAL_CODE_TYPE',BlackboardAttribu
te.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "PostalCode")
                family_name_att_type =
blackboard.getOrAddAttributeType('BMW_FAMILY_NAME_TYPE',BlackboardAttribu
te.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "FamilyName")

                attributes = ArrayList()
```

```
attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_USER_ID.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName,
Contact_ID))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_NAME.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName, GivenName))

attributes.add(BlackboardAttribute(family_name_att_type,CicIviBmwDbIngest
ModuleFactory.moduleName, FamilyName))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_LOCATION.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName,
StreetHousenumber))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_CITY.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName, City))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_COUNTRY.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName,
Country))

attributes.add(BlackboardAttribute(postal_code_att_type,CicIviBmwDbIngest
ModuleFactory.moduleName, Postalcode))



                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(

ModuleDataEvent(CicIviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_CONTACT, None))



        #database contactbook search bluetooth

        files = fileManager.findFiles(dataSource, "contactbook_2010%.db")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
```

```
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db")
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the bluetooth table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT Bt_ID,
HEX(Bt_address) as hexBt from bluetooth")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    Bt_ID = resultSet.getString("Bt_ID")
                    self.log(Level.INFO, "Bt_ID" + Bt_ID)
                    Bt_address = resultSet.getString("hexBt")
                    self.log(Level.INFO, "Bt_address" + Bt_address)

                    # Make an artifact on the blackboard,
TSK_BLUETOOTH_PAIRING and give it attributes for each of the fields
                    art =
file.newArtifact(BlackboardArtifact.ARTIFACT_TYPE.TSK_BLUETOOTH_PAIRING)
                    bluetooth_address_att_type =
blackboard.getOrAddAttributeType('BMW_BLUETOOTH_ADDRESS_TYPE',BlackboardA
ttribute.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "Bt_address")

                    attributes = ArrayList()

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_ID.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName, Bt_ID))

attributes.add(BlackboardAttribute(bluetooth_address_att_type,CicIviBmwDb
IngestModuleFactory.moduleName, Bt_address))

                    art.addAttributes(attributes)
                    try:

Case.getCurrentCase().getSleuthkitCase().getBlackboard().postArtifact(art
, CicIviBmwDbIngestModuleFactory.moduleName)
                    except Blackboard.BlackboardException as e:
                        self.log(Level.SEVERE, "Error posting artifact "
```

```
+ art.getDisplayName())

                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")



        #mme database
        #mediastores

        files = fileManager.findFiles(dataSource, "mme")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db" )
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the mediastores table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT msid, capabilities,
mssname, name, identifier, mountpath FROM mediastores")

                # Cycle through each row and create artifacts
                while resultSet.next():
                    try:
                        msid = resultSet.getString("msid")
                        capabilities = resultSet.getLong("capabilities")
                        mssname = resultSet.getString("mssname")
                        name = resultSet.getString("name")
                        identifier = resultSet.getString("identifier")
                        mountpath = resultSet.getString("mountpath")
```

```
                    except SQLException as e:
                        self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")

                    # Make an artifact on the blackboard, TSK_DEVICE_INFO
and give it attributes for each of the fields
                    art =
file.newArtifact(BlackboardArtifact.ARTIFACT_TYPE.TSK_DEVICE_INFO)
                    attributes = ArrayList()

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_ID.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName, msid))
                    timevalue = capabilities/1000000000

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_DATETIME.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName,
timevalue))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_DESCRIPTION.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName,
mssname))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_NAME.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName, name))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_DEVICE_ID.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName,
identifier))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_PATH.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName, mountpath))

                    art.addAttributes(attributes)

                    try:
                        # index the artifact for keyword search
                        blackboard.indexArtifact(art)
                    except Blackboard.BlackboardException as e:
                        self.log(Level.SEVERE, "Error indexing artifact "
+ art.getDisplayName())

                # Fire an event to notify the UI and others that there
are new artifacts
                IngestServices.getInstance().fireModuleDataEvent(

ModuleDataEvent(CicIviBmwDbIngestModuleFactory.moduleName,
                    BlackboardArtifact.ARTIFACT_TYPE.TSK_DEVICE_INFO,
None))
            except SQLException as e:
                self.log(Level.INFO, "Error querying database "
+file.getParentPath()+file.getName() + " table (" + e.getMessage() + ").
Ignoring this file and continuing ingest.")
                #Miguel: esta instrucao deixa de ser necessaria.
                #return IngestModule.ProcessResult.OK



        #mme database
        #software info
        files = fileManager.findFiles(dataSource, "%mme_custom")
```

```
        artIId = blackboard.getOrAddArtifactType("TSK_SOFTWARE_INFO",
"Software info")
        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db" )
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the software info table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT version FROM
software_info")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    version = resultSet.getString("version")



                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")

                # Make an artifact on the blackboard, TSK_SOFTWARE_INFO
and give it attributes for each of the fields
                art = file.newArtifact(artIId.getTypeID())
                attributes = ArrayList()


attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_VERSION.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName,
```

```
version))

                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(

ModuleDataEvent(CicIviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_DEVICE_INFO, None))


        #mme library folders
        files = fileManager.findFiles(dataSource, "%mme_library")

        artIIIId = blackboard.getOrAddArtifactType("TSK_FOLDERS",
"Folders")
        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db" )
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the library folders table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT foldername,
last_sync, basepath FROM folders")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
```

```
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    foldername = resultSet.getString("foldername")
                    last_sync = resultSet.getLong("last_sync")
                    basepath = resultSet.getString("basepath")



                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")

                # Make an artifact on the blackboard, TSK_FOLDERS and
give it attributes for each of the fields
                art = file.newArtifact(artIIIId.getTypeID())
                folder_name_att_type =
blackboard.getOrAddAttributeType('BMW_FOLDER_NAME_TYPE',BlackboardAttribu
te.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "foldername")

                attributes = ArrayList()


attributes.add(BlackboardAttribute(folder_name_att_type,CicIviBmwDbIngest
ModuleFactory.moduleName, foldername))
                timevalue = last_sync/1000000000

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_DATETIME_MODIFIED.getTypeID(),
CicIviBmwDbIngestModuleFactory.moduleName, timevalue))

attributes.add(BlackboardAttribute(BlackboardAttribute.ATTRIBUTE_TYPE.TSK
_PATH.getTypeID(), CicIviBmwDbIngestModuleFactory.moduleName, basepath))


                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(

ModuleDataEvent(CicIviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_DEVICE_INFO, None))



        #Library albuns

        files = fileManager.findFiles(dataSource, "mme_library%")

        artIIIIId = blackboard.getOrAddArtifactType("TSK_LIBRARY_ALBUNS",
"Library albums")
```

```
        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db" )
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the Library albuns table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT
library_albums.album FROM library_albums")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    album = resultSet.getString("album")


                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")

                # Make an artifact on the blackboard, TSK_LIBRARY_ALBUNS
and give it attributes for each of the fields
                art = file.newArtifact(artIIIIId.getTypeID())
                library_albums_att_type =
blackboard.getOrAddAttributeType('BMW_LIBRARY_ALBUMS_TYPE',BlackboardAttr
ibute.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "album")
                attributes = ArrayList()


attributes.add(BlackboardAttribute(library_albums_att_type,CicIviBmwDbIng
estModuleFactory.moduleName, album))
```

```
                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(

ModuleDataEvent(CicIviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_METADATA, None))


        #Library artists
        files = fileManager.findFiles(dataSource, "%mme_library%")
        artIIIIIId =
blackboard.getOrAddArtifactType("TSK_LIBRARY_ARTISTS", "Library artists")

        numFiles = len(files)
        progressBar.switchToDeterminate(numFiles)
        fileCount = 0
        for file in files:

            # Check if the user pressed cancel while we were busy
            if self.context.isJobCancelled():
                return IngestModule.ProcessResult.OK

            self.log(Level.INFO, "Processing file: " + file.getName())
            fileCount += 1

            # Save the DB locally in the temp folder. use file id as name
to reduce collisions
            lclDbPath =
os.path.join(Case.getCurrentCase().getTempDirectory(), str(file.getId())
+ ".db" )
            ContentUtils.writeToFile(file, File(lclDbPath))

            # Open the DB using JDBC
            try:
                Class.forName("org.sqlite.JDBC").newInstance()
                dbConn = DriverManager.getConnection("jdbc:sqlite:%s"  %
lclDbPath)
            except SQLException as e:
                self.log(Level.INFO, "Could not open database file (not
SQLite) " + file.getName() + " (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK

            # Query the Library artists table in the database and get all
columns.
            try:
                stmt = dbConn.createStatement()
                resultSet = stmt.executeQuery("SELECT
library_artists.artist FROM library_artists")
            except SQLException as e:
                self.log(Level.INFO, "Error querying database for
contacts table (" + e.getMessage() + ")")
                return IngestModule.ProcessResult.OK
```

```
            # Cycle through each row and create artifacts
            while resultSet.next():
                try:
                    artist = resultSet.getString("artist")


                except SQLException as e:
                    self.log(Level.INFO, "Error getting values from
contacts table (" + e.getMessage() + ")")

                # Make an artifact on the blackboard, TSK_LIBRARY_ARTISTS
and give it attributes for each of the fields
                art = file.newArtifact(artIIIIIId.getTypeID())
                library_artists_att_type =
blackboard.getOrAddAttributeType('BMW_LIBRARY_ARTISTS_TYPE',BlackboardAtt
ribute.TSK_BLACKBOARD_ATTRIBUTE_VALUE_TYPE.STRING, "artist")
                attributes = ArrayList()


attributes.add(BlackboardAttribute(library_artists_att_type,CicIviBmwDbIn
gestModuleFactory.moduleName, artist))

                art.addAttributes(attributes)
                try:
                    # index the artifact for keyword search
                    blackboard.indexArtifact(art)
                except Blackboard.BlackboardException as e:
                    self.log(Level.SEVERE, "Error indexing artifact " +
art.getDisplayName())

            # Fire an event to notify the UI and others that there are
new artifacts
            IngestServices.getInstance().fireModuleDataEvent(

ModuleDataEvent(CicIviBmwDbIngestModuleFactory.moduleName,
                BlackboardArtifact.ARTIFACT_TYPE.TSK_METADATA, None))



        #Post a message to the ingest messages in box.
        message =
IngestMessage.createMessage(IngestMessage.MessageType.DATA,
            "Sample Jython Data Source Ingest Module", "Found %d files" %
fileCount)
        IngestServices.getInstance().postMessage(message)

        return IngestModule.ProcessResult.OK
```