# Overview

## Source structure

* src
  + Linux
  + Unix
  + Win32

## Design

* Interfaces
  + DebuggerInterface
    - attach
    - detach
    - g
    - bpx
    - bpl
    - bpc
    - bpd
    - bpe
    - r
  + GUIDisplayInterface
    - hexDisplay(address, length)
    - mapDisplay(address, length, colorMap)
  + MemReaderInterface
    - readAddr(addr)
    - readQword(addr) # Also accessible by dq
    - readDword(addr) # Also accessible by dd
    - readWord(addr) # Also accessible by dw
    - readByte(addr) # Also accessible by db
    - readMemory(addr, length)
    - readString(addr, isUnicode)
    - getPointerSize()
    - getEndianity()
  + MemWriterInterface
    - writeAddr(addr, value)
    - writeQword(addr, value)
    - writeDword(addr, value)
    - writeWord(addr, value)
    - writeByte(addr, value)
    - writeMemory(addr, data)
* RecursiveFind
  + recursiveFind(target, start\_address, length, hops, delta, must, isVerbos)
  + printRecursiveFindResult
* MemReaderBase
  + resolveOffsetsList
  + readNPrintQwords(addr, length, isNoBase, itemsInRow)
  + readNPrintDwords(addr, length, isNoBase, itemsInRow)
  + readNPrintWords(addr, length, isNoBase, itemsInRow)
  + readNPrintBin(addr, length, isNoBase, itemsInRow)
* Utile
  + DATA
  + makeQwordsList, makeDwordsList, makeWordsList, makeBytesList
  + printIntTable
  + printAsQwordsTable, printAsDwordsTable, printAsWordsTable
  + hex2data, data2hex, hex2dword, dword2hex
  + buffDiff(buffers, chunk\_size)
  + dotted
  + getIpcsInfo(isVerbos)
  + getAllShmidsInfo(ownerFilter)
  + getShmids(ownerFilter)
* Patterns
  + CreatePatternsFinder(memReader)
  + PatternFinder
    - Search(pattern, startAddress, lastAddress, context)
  + SearchContext
  + displaySearch
  + verbosSearch
  + GetOffsetByName
  + SHAPE
    - \_\_init\_\_(name, place, data, extraCheckFunction=None, fromStart=False)
    - getData
    - isValid
    - setForSearch(self, patFinder)
    - size
    - getAlignment
    - *getName (do not orverwrite)*
    - *getPlace (do not overwrite)*
    - *getValidRange (do not overwrite)*
  + ANYTHING(size=0)
  + POINTER(isNullValid=False, valueRange=None)
  + NUMBER(value=None, size=None, alignment=None, isSigned=False, endianity=’=’)
  + BYTE
  + WORD
  + DWORD
  + QWORD
  + BUFFER(size)
  + STRING(size=None, maxSize=0x1000, fixedValue=None, isPrintable=True, isUnicode=False, isCaseSensitive=True)
  + STRUCT(content)
  + ARRAY(size, var)
  + POINTER\_TO\_STRUCT(pattern, isNullValid=False)

## Implementations

* Win32
  + MemReaderBaseWin
    - enumModules
    - findModule
    - getAllSections
    - findSection
    - getHandles
  + Debugger
  + MemoryReader
  + ExternalMemoryReader
    - Uses memReader.exe
  + DifferentialSearch (<http://nativassaf.blogspot.com/2011/11/map-hack-tutorial_15.html>)
    - readAllMemoryWithAttributes
    - removeChangedMemory
    - removeUnchangedMemory
    - searchUInt32
    - searchInt32
    - searchUInt16
    - searchInt16
    - searchUInt8
    - searchInt8
    - \_\_len\_\_
  + InjectDll
    - inject(process\_id, dllName, LoadLibraryA\_address, isVerbos)
    - createProcessWIthDll(cmdLine, dll, …)
* Unix
  + MemoryReaderCommunicator implements MemReaderBase
    - Attach(memInfo, pointerSize, defaultSize)
      * memInfo is list of tuples of shmid, baseAddr and size
    - Uses memReader.cpp
* Linux
  + SharedMemReader
    - Attach(memInfo)
      * memInfo is list of tuples of shmid, baseAddr and size

## Common errors

* *AttributeError: 'DWORD' object has no attribute 'minOffset'*Data type is used where a range is needed, meaning replace DWORD with a simple tuplet.
* *TypeError: \_\_init\_\_() got multiple values for keyword argument 'size'*Size keyword was used with a QWORD / DWORD / WORD / BYTE data types. Use NUMBER if you want to set the data size yourself.