Extending IDA with Scripts

Dr. Jared DeMott
CHIEF HACKING OFFICER

@jareddemott www.vdalabs.com

Overview

IDC

IDApy

SDK

IDC Scripting

C like

- No data types

Declare Variables at Top of Script

- No globals
- Arrays are not fun

From IDA help

- IDC Language
  - Expressions
  - Statements
  - Variables
  - Functions

IDC

Variables

- Use ‘auto’ as the key word

Functions

- All are defined with the "static"
- Argument list does not require any type info or the auto keyword
- Return type never specified

Example IDC Function

```
static myCoolFunc(arg1, arg2) {
    auto var1;
    var1 = arg1 + arg2;
    return var1;
}
```

IDC

Statements

- Most C available
  - Loops
    - For, while, do. Break/continue
      - {} same as C
- No switch or goto

Expressions which Differ from C

- Integers promote to float as required
- Add (‘+’) string concatenation
- Comparisons work for string operands
  - If (“frank” == “john”)

Interacting with IDA Database

Read Data

- Return -1 if virtual addr is invalid
- long **Byte**(long addr);
- long **Word**(long addr);
- long **Dword**(long addr);

Writing to IDA DB

- Useful for self-mod code
- void **PatchByte**(long addr, long val);
- void **PatchWord**(long addr, long val);
- void **PatchDword**(long addr, long val);

Interacting with Reverse Engineer

void Message (string format, ...)

- Print to the message area
  - We'll use "print" in python

void Warning (string format, ...)

- Show warning dialog box

long AskYN (long default, string prompt)

- Ask a yes/no in dialog box

string AskFile (bool forsave, string mask, string prompt)

Cursor Control and Address Operations

long ScreenEA()

- Read current cursor location
  - Return virtual address

long Jump(long addr)

- Set cursor to indicated location

long LocByName(string name)

- Return address for specified name

long Name(long ea)

- Return name of virtual address

Iterating Cross References

Iterate Code

- Rfirst, Rnext, RfirstB, RnextB
  - B is “to”; other is “from”

Iterate Data

- Dfirst, Dnext, DfirstB, DnextB

Iterate Segments

- FirstSeg, NextSeg

Iterate Functions

- long **NextFunction**(long ea)
- long **PrevFunction**(long ea)
- long **NextHead**(long ea, long max)
  - Return the addr of the next item following the ea

```
#include <idc.idc>

static main() {
    list_callers("strcpy");
    list_callers("sprintf"); // use _name for some compilers
}
```

```
static list_callers(bad_func) {
    auto func, addr, xref, source;
    func = LocByName(bad_func);
    if( func == BADADDR) {
        Warning("Sorry, %s, not found in database", bad_func);
    }
    else {
        for( addr = RfirstB(func); addr != BADADDR; addr = RnextB(func, addr)) {
            xref = XrefType();
            if( xref == fl_CN || xref == fl_CF) {  //function calls only (near call or far call)
                source = GetFunctionName(addr);
                Message("%s is called from 0x%x in %s\n", bad_func, addr, source);
 } } } }
```

IDA python

Extend IDA

- Plugin that allows scripts to be authored in Python
- Scripts have access to IDA and Python APIs
- Common amongst security researchers and malware analysists

Demo

Decode the secret message

- Pretend we're working with live malware that we don't want to actually run/execute
  - Find key
  - Reverse decoding algorithm
  - Write python to emulate that
  - Print the decoded message to the output window
- *1_skeleton.py → 2_skeleton.py*

SDK Extensions

Exploring Plugins

Example Extension

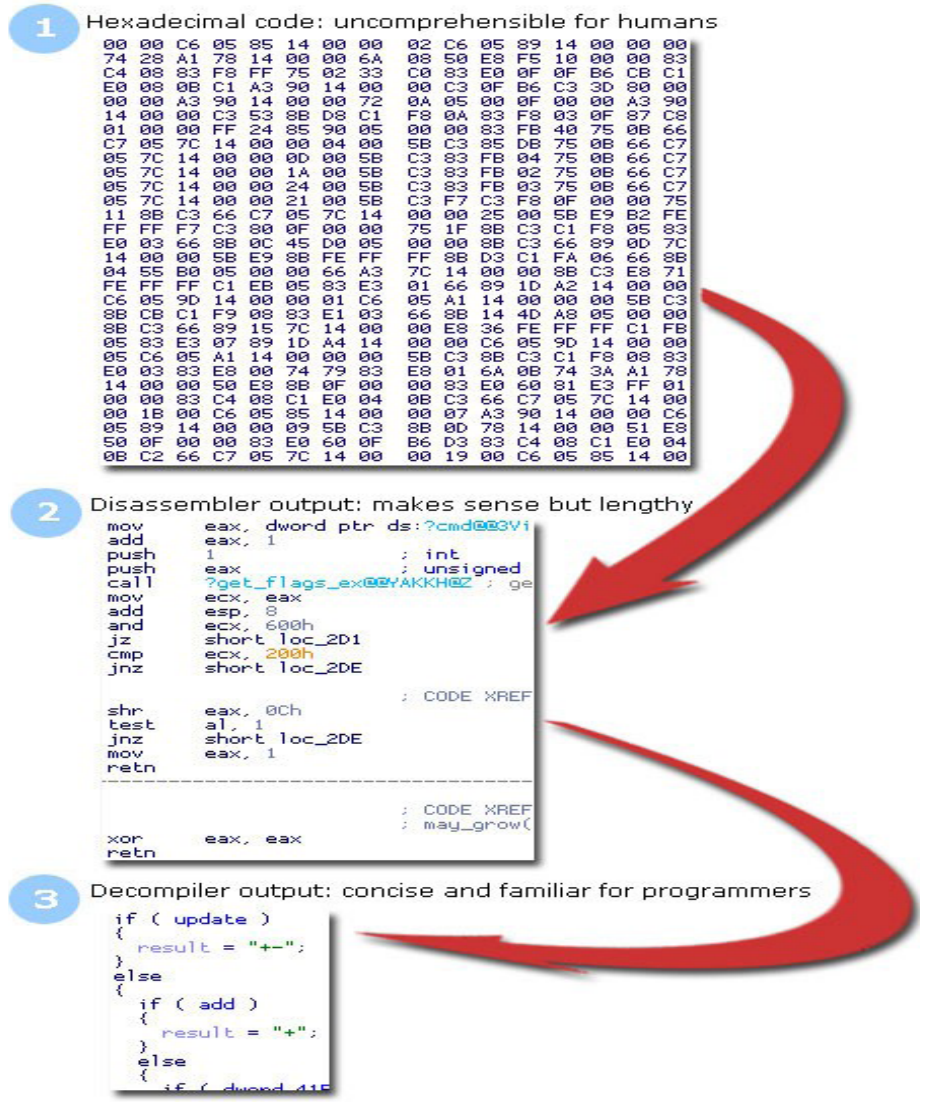
CollabREate – Chris Eagle & Tim Vidas

- http://www.idabook.com/collabreate
- IDA Plugin is client
  - Copy the .plw into plugin dir
    - Must be compiled for version of IDA
- Java server w/ Postgres/Mysql backend
  - Commands are intercepted as they occur and sent to the server
  - Server caches and distributes commands
  - Authentication and project management

http://hex-rays.com

Pseudo code in IDA

Might not Properly Decompile Malware



IDA SDK

Required to Build Full Extensions

- Written in C++
- A .dll or shared object in UNIX
- Steeper learning curve
  - Worth it when more power is required

Three Types of Extensions can be Created

- Plug-in
  - Extend IDA functionality
- Processor Module
  - Extend IDA to understand new instruction sets
- File Loader
  - Extend IDA to understand new executable/object file formats

SDK

Layout and API

- *Libxxx directories contain IDA libs for building plugins/loaders/modules with various compilers*
- *Bin* directory contains compiled plugins
- Include directory contains header files
- *Ldr* directory contains sample loaders for file formats
- *Module* directory contains sample processor modules
- *Plugins* directory contains sample plugins

Plugin Architecture

Plugins Export

- *plugin_t* PLUGIN
  - Defined in loader.hpp

Describes Plugin Options

- Name of the init function
- Name of the terminate function
- Name of the run function
- Desired hotkey to activate the plugin

More

http://www.openrce.org/reference_library/files/ida/idapw.pdf

Also

- IDC to SDK Reference
  - At the end of Eagle's book
- You can also use Windows API in plugin
  - http://www.openrce.org/reference_library/ida_sdk

Summary

Python Scripting in IDA

SDK

Next:

- Course on Exploit Development