Security Research and Development Framework (SRDF)

By Amr Thabet

Abstract:

This is a free open source Development Framework created to support writing security tools and malware analysis tools. And to convert the security researches from the theoretical approach to the practical implementation.

This development framework created mainly to support the malware field to create malware analysis tools and anti-virus tools easily without reinventing the wheel and inspire the innovative minds to write their researches on this field and implement them using SRDF.

Introduction:

In the last several years, the malware black market grows widely. The statistics shows that the number of new viruses increased from 300,000 viruses to millions and millions nowadays.

The complexity of malware attacks also increased from small amateur viruses to stuxnet, duju and flame.

The malware field is searching for now technologies and researches, searching for united community can withstand against these attacks. And that's why SRDF

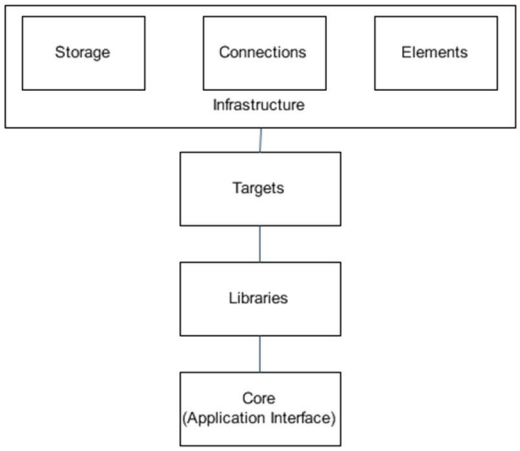
The SRDF is not and will not be developed by one person or a team. It will be developed by a big community tries to share their knowledge and tools inside this Framework

SRDF still not finished … and it will not be finished as it's a community based framework developed by the contributors. We just begin the idea.

The SRDF is divided into 2 parts: User-Mode and Kernel-Mode. And we will describe each one in the next section.

The User-Mode Part:

The Design:



Infrastructure:

This includes the essential elements of any development framework and it's not related to security like: string, hash, list, serializer, database, registry manipulation, sockets and so on.

We decided to create this part rather than depending on any development framework to make this framework independent from any other development frameworks and to be portable on any development framework

Targets:

This is the beginning of the SRDF. This part is related to security and it's simply the

Target from your security. What do you want to secure or secure from. And it includes Files (PE Files and others), Processes and Packets.

Libraries:

That's the security tools that the SRDF support. And it's divided into two namespaces: malware and network

Malware includes the assemblers and disassemblers, emulator, debugger, Yara Scanner (wildcard scanner) file recursive scanner and other tools

Network includes User-Mode and Kernel-Mode packet capturing and Firewall

Core (The Application Interface):

The Core includes the main management systems like memory management and thread management. Also, it includes the security coding modules and includes the Logging system and the back-end Database.

And also, it's the Application Interface. As, it's the Application class that you will inherit the whole your application from. And it includes many features to write your application on it.

The Infrastructure:

Elements:

It's divided into three namespaces:

1. **String**: it contains the string class, encoded string, hash and list
2. **Code:** it contains the NativeCode class and StoredProcedure … and they represents the shellcode and the code that stored in database. Like a virus detection routines inside an Antivirus
3. **XML:** and it contains the XML Encoder and the Serializer.

Connections:

It's divided into three namespaces:

1. **Internet:** and it contains the internet communication protocols like sockets, HTTP Sockets and so on.
2. **IPC:** and it contains the Inter-Process Communication protocol
3. **User to Kernel Mode Communication:** and it contains the communication protocol to communicate to the kernel-mode part of the SRDF

Storage:

It's divided into three namespaces:

1. **Databases:** and it contains the Database class and SQLiteDB and so on.
2. **Files:** and contains the File writing and logging classes
3. **Registry:** and it contains the registry read and write

The Targets:

Files:

This namespace describes the File Formats of The Files that could contain malicious code like: Executable Files (PE and ELF) and Document Files (PDF, Docx …) and so on.

Until now it contains The PE/PE+ Files parsers

Process:

And it includes one class only named cProcess. And, this class describes a running process and parses its PEB and gives you the important information about the process. And support injecting code and create a remote thread.

Packets:

And it includes classes that describe an internet packets captured on the wire or generated for an attack.

Libraries:

It contains two namespaces:

Malware:

This namespace contains the scanning, Hooking and emulation libraries and contains Pokas Emulator wrapper class, Yara wrapper class (wildcard scanner), a debugger and contains a directory recursive scanner and other tools

And also, it contains the x86 assembler and disassembler (using Pokas Emulator Assembler) and allow to contain other assemblers and for other platforms.

Network:

This namespace contains the User-Mode Packet capture and firewall. And contains the Winpcap Packet capturing and firewall system.
It also should include Application Layer parsers for FTP, HTTP, IRC and all known protocols and include Pcap Reader and writer.

The Core:

And the core includes the cApp class that contains the management systems for memory and threads. And includes the back-end database and logging

Memory Management System:

We decided to create a memory manager to meet these goals:

1. Secure from Heap Overflow
2. Automatic variable initialization
3. Fast
4. Garbage collecting per thread (as per object will need a modification on memory use itself)

So we designed this Memory Manager and make it separate between the buffer header and buffer data without decreasing the performance and to include the Thread Id for each buffer to be freed by the Thread Management system on the termination of each thread

Thread Management System:

It's a simple thread management system designed to meet these goals:

1. Free Local buffers for each terminated thread
2. Limit the thread creation into a specific number of threads
3. Synchronize on database and log

Secure Coding Modules:

And it contains functions and macros for securing your application from buffer overflows, format string vulnerabilities and heap overflow vulnerabilities.

The Kernel-Mode:

The Kernel-Mode Goals:

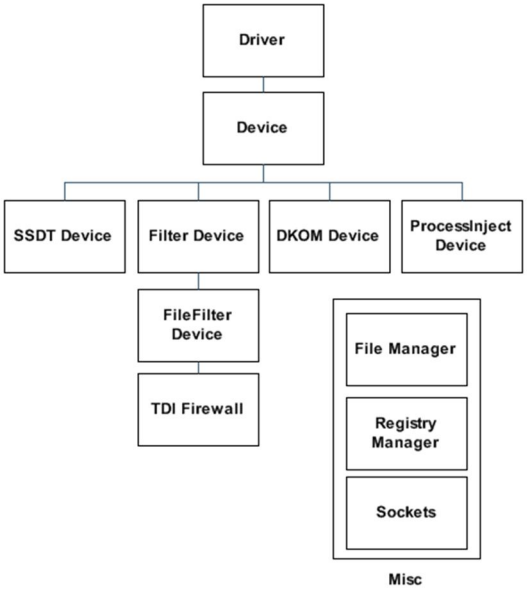
The Goals of the kernel-Mode development Framework are:

1. Easy to create a Kernel-Mode security tool
2. Support OOP using the native device driver programming APIs
3. Support detaching between devices in IRPs
4. Easy to use files, registry and so on
5. Create a User-Mode/Kernel-Mode communication protocol
6. Designed only for hooking and security tools.

The Kernel-Mode SRDF is designed on native device driver programming APIs and independent from the WDF (windows drivers foundation).

Now we will describe the design of Framework and then we will go through the IRP dispatching mechanism in the KM-SRDF

The Design:



Driver: It's the core management system that dispatching the IRPs to the devices and manage the devices.

Device: it represents a device object and it contains the IRP dispatching between the control device object and the filtering device objects and includes attaching and detaching from a devices chain and all necessary functions for a device object

SSDT Device: this class is inherited from device class and it's created for SSDT Hooking

Filter Device: this class created for attaching to a chain and filtering the inputs and the outputs of the IRPs

File Filter Device: this class is inherited from Filter Device and it's created for filtering the File system I/O request packets (IRPs) or monitoring file operations

TDI Firewall: this class is inherited from Filter Device and it's created for filtering the internet packets and connections and the processes that tries to connect to the internet

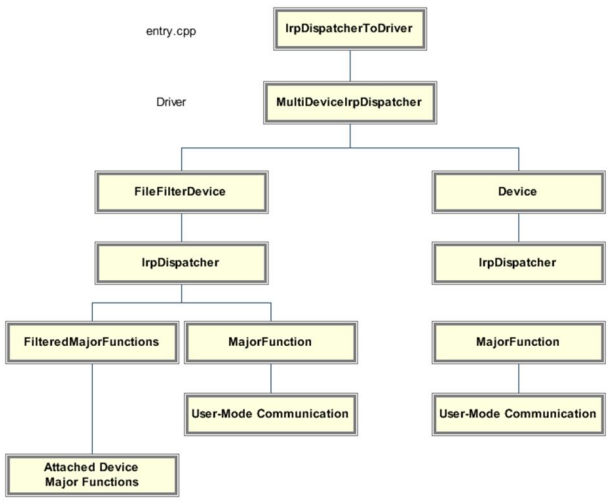
DKOM Device: this class created to provide a generic way to work with opaque structures in windows without worrying about windows version and subversion

Process Inject Device: this class provides a way to inject code or modify the memory of a process from the kernel-mode

File/Registry Managers: they are tools created to support writing files and working with registry easily without worrying about IRQL

Sockets: it's an easy interface to connect to the internet using the TDI interface

The IRP Dispatching:



- The IRP dispatching begins from the entry.cpp and it dispatch the IRP to the Driver

- The driver checks the device object and dispatch the IRP to the related device
- The device sends the IRP to the User-Mode communication object to work with it as it's sent to the control device object
- If it's a FileFilter Device, the device dispatches the IRP based on the device object to the Attached Device Objects or to the control device object and the user-mode communication

To do list:

Until now, not all which described above yet finished … most of them, but not all

And also, we wish to add:

- Static Unpackers Library
- Supporting more file formats (PDF, Elf, Docx, Zip, jar, Android Apps and all necessary file formats)
- Supporting more innovative secure coding modules
- Supporting assemblers, disassemblers and emulators for .Net and Java
- Support shellcode encoders
- Support NDIS, kernel sockets and more new libraries
- More documentations

Source Code:

http://code.google.com/p/srdf/

Conclusion:

This development framework will support the anti-malware technologies to grow and support implementing researches in the malware field more to withstand against the new attacks nowadays

The framework is based on community and we aim to create a big community for it. We didn't finished the framework … we just begin

About the Author:

I'm Amr Thabet. I'm a Freelancer Malware Researcher. I began in this field from nearly 4 years.
I'm the Author of Pokas x86 Emulator. I gave a talk in Cairo Security Camp 2010 and the University of Sydney.