File Disinfection Framework (FDF)

Striking back at polymorphic viruses

REVERSING LABS

CONTENTS

INTRODUCTION

"Invincibility lies in the defense; the possibility of victory in the attack." – Sun Tzu

Polymorphic viruses make up an ever-increasing percentage of daily malware collections. The sophistication of these attacks significantly exceeds the capabilities of existing classification and handling solutions. The situation goes from bad to worse when we attempt the most complicated part of incident response, file disinfection and remediation.

To combat this problem we've created a new open source project, the File Disinfection Framework (FDF), built on top of a new generation of TitanEngine and tailored specifically to aid in solving these hard problems. FDF combines both static analysis and emulation to enable users to rapidly switch between modes of operation to use the best features of each approach. Highly advanced static functions are hidden behind a simple and easy-to-use program interface that enables the broad range of capabilities that are required for decryption, decompression and disinfection. Their complement is a set of functions that enable quick and very customizable emulation. For the first time, analysts will have the ability to truly see and control everything that happens inside the emulated environment. They can run high level code inside the context of the emulated process to influence objects and files and direct the execution flow.

There is a great asymmetry today. Proper sample reversing can take longer than arming or writing of the same. Case in point, custom packed and protected file infector samples and server side kits that apply multiple layers of packing, protection and complex polymorphic infection in real time as samples are pushed to the end point. With FDF, defenders will have a proper toolbox that will allow total binary control and disinfection, thus being able to consolidate their efforts across responding teams and building reusable disinfection components that save time and avoid system re-imaging.

Strategies for defenders are to build a solid toolbox of components and modules that are unified, powerful and capable of integrating knowledge gained from individual analysis that occurs with every manual analysis and every time a specialized disinfection tool is required. In this way, we are fortifying our reversing toolset to better deal with complications, protection, obfuscation and polymorphic infection techniques in an automated manner. This strategy will give exponential returns over time as continuous work on a unified body of tools will be able to automatically handle ever increasing complications that will be invented.

FILE DISINFECTION FRAMEWORK (FDF)

File Disinfection Framework (FDF) is an open source project that implements an advanced virtual machine for polymorphic malware disinfection. It will enable dynamic binary analysis on top of a static analysis framework by giving developers full control over detection, disinfection and repair of affected files.

State of the Art in malware disinfection today falls into two categories: a) generic disinfection routines, and b) targeted disinfection routines. Generic disinfection routines like generic unpackers or generic behavioral signatures show promise but are easily defeated and can rarely disinfect serious polymorphic file infectors such as Sality and Virut.

For example, Sality can unexpectedly increase sizes of infected executable files. It will infect executable files on local, removable and remote shared drives by replacing the original host code at the entry point of the executable to redirect execution to the polymorphic viral code, which has been encrypted and inserted in the last section of the host file. As an Entry-Point Obscuring (EPO) polymorphic file infector, the virus then gains control of the host body by overwriting the file with complex and encrypted code instructions.

On the other hand Virut is a polymorphic appending file infector with EPO capabilities. This malware uses several infection methods:

> a) It relocates a certain amount of bytes from the entry point of the original file and writes its initial decryptor there. So when an infected file is run, the malicious code gets the control first. The initial decryptor then decrypts a small part of the malware's body which is appended to the end of the infected file and passes control to it;
>
> b) The malware appends its code to the end of the file and changes the entry point address of the original program so it points to the start of the appended malicious code, where the decryptor is located. This is the most common way of infecting files for appending parasitic infectors;
>
> c) The malware writes its initial code into a gap (empty space) in the end of the original file's code section and redirects the entry point address to that code. The initial code decrypts a small part of the malware body and passes control to it. Then the main decryptor takes control and decrypts the rest of the virus body. Once the file is infected, malware patches the first found API call (from the entry point address) in the original program so that instead of the API, it calls the initial malware decryptor. That decryptor may then be located in the end of the code section as said above.

Given these examples, to affect a serious disinfection, often including rootkits, targeted routines need to be developed. They are usually available as standalone tools that affected users need to apply using special mechanism or precautions, or need other professionals (e.g. AV company support teams or Geek Squads) to apply. Development of targeted routines is a cumbersome and slow task and hence very few organizations are incentivized to develop them as it is much easier to advocate a full re-install rather system repair. Yet such methods

fall short in Enterprise or Government environments where re-imaging thousands of machines may be prohibitively time consuming and expensive.

Yet those malware analysts or developers tasked with the development of targeted disinfection routines do not have a framework or a set of tools that would help them quickly and accurately develop a targeted disinfection application. They largely write each application from scratch, introducing development delay, large QA cycles, and a greater risk that disinfection remedy may be worse than the infection itself.

FDF proposes to improve this status quo by enabling writers of disinfection tools to quickly identify, disinfect and repair affected binaries by leveraging proven and well tested open source code. In addition to an advanced virtual machine specifically designed to address problems of polymorphic malware disinfection, FDF will enable dynamic and static binary analysis in order to round out a tool necessary for complete file disinfection.

DISINFECTION SOLUTION DEVELOPMENT TODAY

	FDF	**Home grown solutions**
Code reuse	Yes	No
VM for code disinfection	Yes	No
Dynamic analysis APIs	Yes	No
Static analysis APIs	Yes	No
Parallel analysis	Yes	No
Automated file repair	Yes	Only with generic implementations

GOALS

FDF will target polymorphic malware disinfection by implementing three key components:

a) Advanced virtual machine,
b) Dynamic analysis APIs, and
c) Static analysis APIs.

Advanced virtual machine will introduce a controlled execution environment specifically to address execution control during the disinfection process.

Dynamic analysis APIs will benefit from the advanced virtual machine as potentially unwanted flow control will be controlled and developers will not need external emulators or execution within a sandboxed or a fully blown virtual environment. This is exactly what is required to deal with the current generation of polymorphic infectors.

Static analysis APIs will introduce a new collection of static functions that manipulate the executable content without a need for further emulation infrastructure. By definition such APIs are more complex to use, but afford significant performance gains and execution safety.

In addition, FDF will enable two key automation features:

- Parallel analysis of multiple files
- APIs for automated file repair

TARGET AUDIENCE

Development of FDF will leverage the TitanEngine user base and allow a wider community of reversing engineers to tackle complex file disinfection projects more efficiently and effectively. If we are successful in this effort as we expect to be, many organizations (AV companies, individual researchers and non-profits such as CERTs) will be investing additional resources in the development of complex disinfection modules rather than relying on commercial vendors who have historically been unable to deliver prompt responses to new threats. Hence modules developed with FDF would improve security responses for a wide variety of affected users, from consumers and enterprises, all the way to governmental institutions.

FEATURES

FDF is an open source framework for polymorphic malware disinfection. It implements an advanced virtual machine and enables dynamic and static disinfection of infected binaries. It leverages TitanEngine, a strong open source library for dynamic and static manipulation of executable code. FDF implements the following five key components:

- Advanced Virtual Machine for polymorphic malware disinfection
- APIs for automated file repair
- Dynamic disinfection APIs
- Static disinfection APIs
- Collection of sample applications and documentation

Given that the quantity and complexity of threats has been growing rapidly in the last few years, it is imperative that our reversing tools grow in sophistication, complexity and maturity. File disinfection is the final frontier where the gulf between effort invested to infect files and effort invested to disinfect the files is the greatest.

FDF will be a comprehensive solution for file disinfection, giving the practitioners necessary functions required to manipulate various aspects of executable content in memory and on the disk, removing infected content, repairing files and thus reducing development and testing time along with the associated risk of low level executable content manipulation.

WHAT IS NEW

This approach will succeed because the underlying premises for FDF's collection of dynamic and static APIs contained in, TitanEngine have already been proven to work. TitanEngine's APIs have been successfully used by numerous anti-malware, vulnerability and penetration testing researchers. The novel approach with FDF is to develop an advanced virtual machine and a special set of additional dynamic and static APIs that are geared towards file disinfection and add necessary automation that would allow for modern disinfection utilities to be written.

USE OF TITANENGINE

FDF will leverage TitanEngine, a 400+ function strong open source library that was originally released at Black Hat Las Vegas in 2009. TitanEngine was designed as a software project for reversing engineers. It was revolutionary in that several years of experience in writing unpacking (file decomposition) modules was leveraged to realize that there needs to be a logical core, akin to a "reversing" operation system that manipulates executable code. As a result, 400+ necessary functions were extracted in order to implement almost all binary operations on unknown binary content. These are the functions that most reversers kept rewriting and re-implementing in their own projects.

Since its release, TitanEngine has been widely used among anti-malware companies, government labs, vulnerability researchers and penetration testers to easily manipulate and reconstruct binary Windows objects in memory and if necessary convert them into executable

components for subsequent analysis. The most advanced use of TitanEngine was as a building block for the construction of complex format unpackers and for writing of complex disinfection applications. The point of TitanEngine is to speed up creation of reversing software modules by utilizing a core set of 400 well-formed and reliable functions that encapsulate all that a developer needs when manipulating executable content. As such, TitanEngine is a great accelerator for developing FDF where special attention needs to be given to the safe manipulation and repair of targeted executable content.

The solution will give a breathing space to those reversers that have been continually building complex disinfection modules from scratch. Yet more importantly, FDF will be continually expanded to provide one solid cross-platform library for any executable code analysis and disinfection thus avoid fragmentation by practitioners and solving the problem of managing thousands of scripts across many different computing platforms. FDF will be an open source project thus taking away proprietary (and usually poorly tested) tools from the exclusive domain of leading anti-malware labs.

Consolidation of basic reversing blocks for the manipulation of PE content reduces the need of individual practitioners to manage their own (or their organization's) legacy code. Thus they increase their own productivity, response time and insight over attacking code. Open sourcing FDF and relying on community feedback and contributions promotes the lowest cost solution that is spread across a very wide community of practitioners and use case scenarios. As malware development has proven to be very dynamic so should the response tools and techniques.

FDF is a cohesive tool in a responder's arsenal as it consolidates various point solutions and as such allows responders to dramatically increase their response, analysis and disinfection times. This in turn leads to the negation of efforts among adversaries in two key areas: a) custom packing and protection and multi-layering of packing, protection and installation methods and, b) file infection/modification of all executable content on the target system. These two "obfuscation" methods are used by over 80% of unknown and potentially malicious samples today and will need to be much more complex, hence expensive to implement and cheaply distributed, and will need to start leveraging advanced virtualization and evasion techniques that are not easy to implement, require significant computing power and can easily detected.

Today at least 40% of known malicious samples are broken, so-called crapware. FDF will help with the handling and manipulation of such file infecting samples. The next sets of complications are packed and protected samples that often utilize old or obscure encryption, encoding or compression techniques. Because the industry has largely given up on in-depth reversing of unknown binaries, bizarrely these techniques are still used today to delay and evade detection and binary analysis and hence are a great target for FDF, thus enabling creation of simple and powerful disinfection modules. FDF's goal is to address the bulk of file infection methods. But what it will not do today, and which is a work that needs to be augmented in the future, is to support state of the art virtualization protections where multiple levels of indirection (virtual machines, virtual processors, randomized instruction sets) are very tedious and time consuming to reverse and automate in a way that could provide meaningful file disinfection.

Key benefactors of this technology are *Anti-Malware Researchers and Incident Response professionals*. People that work for major AV companies, research institution or staff response teams for Governmental organizations and Enterprises. They write modules that remove protection and obfuscation for families of polymorphic file infectors and packers. There is significant demand to obtain effective utilities which can de-obfuscate and remove certain families of attacking code without requiring full machine re-imaging.

FEATURE LIST

File disinfection framework features:

1. Static analysis functionality that has the ability to view, modify and build on-the-fly PE32/PE32+ files, fields and tables. A large number of embedded decompression routines is included along with systems that dynamically define static structures and build polymorphic decrypters.
2. Highly advanced PE32/PE32+ file validation and repair functionality that completely solves the issues brought up by our last year's BlackHat presentation titled "Constant insecurity: Things you didn't know about PE file format". These functions accurately detect and identify all purposely-malformed PE files that break current security tools or evade detection. In addition, if the file is damaged (as usually happens during virus infections) and deemed repairable, it is automatically repaired to maximize the number of remediated files.
3. Integrated hash database functionality that helps to resolved the otherwise unsolvable problem of reverting function name hashes back to their original names. This custom database is easily extended to add even more libraries and functions to its known hash lists.
4. A truly unique x86 emulator written from scratch that supports the following Windows features
    a. Multiple processes in parallel each in a separate emulated OS
    b. Vital Windows structures: PEB, TEB (with multiple threads) and SEH
    c. x86 assembly code execution with support for FPU and MMX instructions
    d. Windows objects such as handles, mutexes and environment variables
    e. Hundreds of standard Windows APIs that can easily be extended by the user
    f. Dynamically build libraries that mirror the application requirements
    g. The entire file system with customizable drives
    h. Interface which matches the standard Windows debug API
    i. Use of emulated APIs which are directly exposed to user
5. User can call standard Windows APIs inside the context of an emulated process. For example the user can dynamically create a new DLL file inside the virtual file system and load it into the context of an emulated process by calling LoadLibrary equivalent. Every emulated API is exposed to the user and therefore usable with the option of hooking any API one or more times.
    a. Advanced breakpoint logic which includes breakpoints on specific instruction groups and specific instruction behavior such as read or write to a specific part of the memory
    b. Seamless switching between emulation and static analysis
6. Specific functionally designed to disinfect files infected with polymorphic viruses such as Virut and Sality with examples that show its use.

7. Tools to aid in writing disinfection routines such as automatic binary profiling with search for the presence and location of the virus stub.

FUTURE DIRECTIONS

File disinfection framework has been developed under the cyber fast track program run by DARPA and built on top of the new generation of TitanEngine. It's an open source cross platform x86-x64 library that enables its user to unpack, disinfect and build PE32/PE32+ files. These and all Emulation components of the new major release of this framework have been designed to be presented as a BlackHat exclusive. This talk will be followed by the public release of the source code along with whitepapers and sample implementations that outline possible use case scenario for this technology.