Farm: An automated malware analysis environment

Conference Paper · November 2008

DOI: 10.1109/CCST.2008.4751322 · Source: IEEE Xplore

4 authors, including:

Keith B. Vanderveen
Sandia National Laboratories
16 PUBLICATIONS **64** CITATIONS

SEE PROFILE

FARM: AN AUTOMATED MALWARE ANALYSIS ENVIRONMENT

Jamie Van Randwyk, Ken Chiang,
Levi Lloyd, Keith Vanderveen
Sandia National Laboratories[1]
Livermore CA
{jvanran, kchiang, llloyd, kbvande}@sandia.gov

Abstract – We present the Forensic Analysis Repository for Malware (FARM), a system for automating malware analysis. FARM leverages existing dynamic and static analysis tools and is designed in a modular fashion to provide future extensibility. We present our motivations for designing the system and give an overview of the system architecture. We also present several common scenarios that detail uses for FARM as well as illustrate how automated malware analysis saves time. Finally, we discuss future development of this tool.

Index Terms – Computer systems security and privacy, Network intrusion detection technology and virus protection

INTRODUCTION

Malware (malicious software) attacks our information infrastructure from many vectors: via network protocol exploits, attached to media files (Microsoft Word documents, presentations, audio files, etc.), and via malicious download links or scripts on otherwise benign Web pages. Malware includes such malicious code as worms, viruses, Trojan horses, and spyware. They typically get installed on a victim's machine without their knowledge and pose a threat to any organizations interested in protecting their information. To understand the threat and better protect the systems in our network, analysis of the malware is usually required. Since malware is almost always only available in binary form, current approaches to understanding malware capabilities and behaviors rely on reverse engineering (RE), a manually intensive and time-consuming process. RE includes the fields of both static and dynamic analysis, network traffic analysis, and behavioral analysis. Competency in each of these fields requires a dedicated analyst with specific skills encompassing reading and understanding of assembly code, proficiency in operating system internals, expertise in network protocols, and other deep systems knowledge. These extraordinary requirements for the reverse engineer and the ensuing lack of properly trained personnel capable of doing RE, together with the unchecked proliferation of new malware, has made the reverse engineer's time a very precious resource.

We are developing FARM, the Forensic Analysis Repository for Malware, to make the best possible use of scarce human time spent performing quality RE and malware analysis. FARM integrates open source and commercial analysis tools into an automated system in order to reduce the time spent by analysts screening new malware samples. FARM assists analysts in two ways: first, FARM allows analysts to automate an entire series of RE steps previously performed manually; second, automated analysis more quickly points the analyst toward new or more interesting malware, focusing precious RE resources on the most significant threats. Additionally, FARM makes some RE tools and techniques accessible to security analysts who lack RE training, thereby broadening the base of personnel who can participate in malware analysis.

FARM's modular, closed-loop design allows additional analysis tools to be easily integrated into the FARM framework, as well as allowing re-introduction of prior analysis results on a particular piece of malware back into the system for deeper inspection. This paper describes the FARM system's architecture and advantages over traditional RE. Additionally, we will provide case studies on selected pieces of malware in which we have used FARM to aid in analysis. We conclude with proposed work on FARM and a summary.

BACKGROUND & RELATED WORK

Software reverse engineering is a long and difficult process. Few people possess the knowledge and skills required to RE a software binary. Even fewer people can perform RE on binaries that are purposefully obfuscated, as is most modern malware.

A considerable amount of time is required to set up an environment for analyzing malware. For someone lacking experience with RE tools, understanding the various tools and options can be overwhelming. FARM eliminates this step since all of the analyses from the tools are automated, with options set to sensible defaults that can be changed as needed.

Other tools have been developed to automate malware analysis. Some tools, like many anti-virus scanners, only perform static analyses, while others perform only behavioral analyses. Since our tool features both types of analysis modules, we examine several existing tools and discuss their limitations.

CWSandbox [1] is a popular malware behavioral analysis engine that is commercially available. CWSandbox executes malware in an environment which has been instrumented with special API hooks to monitor system API calls.

Norman Sandbox [2] is another behavioral analysis tool that executes malware on virtualized hardware. The virtual environment is isolated from the host operating system and network to prevent accidental contamination. Based on system parameters, Norman determines whether or not system modifications are malicious.

[1] Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Both of these tools are focused mainly on behavioral analysis. Both tools use an environment that is inherently different from a standard setup and can easily be detected by malware, thereby allowing it to evade detection by altering its functionality [3], [4], [5]. Additionally, these tools are constrained to analysis of Windows PE executable type malware.

Threat Expert [6] is an online service that performs static and dynamic analysis of malware submissions. VirusTotal [7] is another online service that performs Anti-Virus (AV) Scans using most well known AV engines. Both online services have the drawback that they require the release of the malware samples to a third party and the functionality is not easily extended to include additional modules.

No single tool offers all the features we needed and provides the level of extensibility desired. Also, none of the aforementioned tools provide automated unpacking which is necessary for static analysis of obfuscated malware.

ARCHITECTURE

One of the design goals of FARM was to architect it in a modular fashion in order to make it flexible and extensible. This is particularly important since analysis tools are continually being created and improved upon. Fig. 1 shows the overall design of the FARM system.

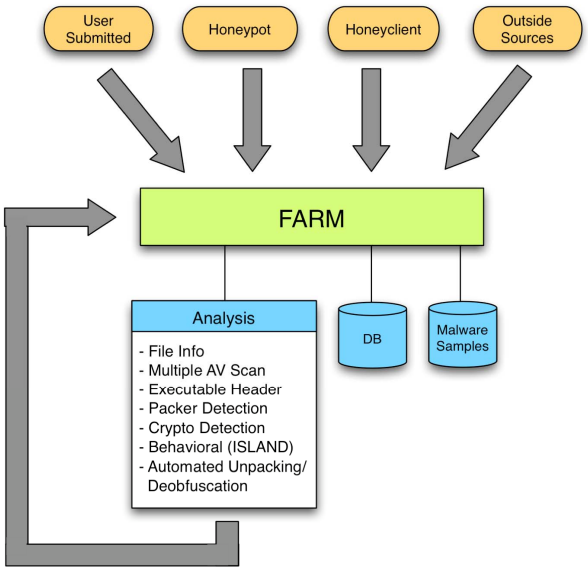

Fig. 1 - FARM Architecture

Malware samples can be introduced into the FARM system by several different means including: user input, honeypots/honeyclients, or other malware collections. FARM archives the malware samples on the system for future retrieval. Once FARM receives a sample, it automatically executes a series of analysis modules and stores the results from the analyses in a relational database. FARM includes a web front-end that enables browsing and searching of malware samples for future reference.

FARM stores meta-data associated with each sample, such as: the source, submission timestamp, and a description. This meta-data is important because it provides useful context for analysts. Also, if an analyst has already done some reverse engineering, the results can be included in the description to help other analysts who may look at the sample in the future.

The first analysis module gathers information about the file. This includes file size, cryptographic hashes, and file type information. The hashes are used to determine whether the sample is unique or if it has been analyzed by FARM previously. If the sample is unique, further analysis takes place; otherwise the new meta-data is linked with the original analysis on the preceding identical sample to eliminate redundant processing and storage. The file type information is also important to the rest of the analysis process because other analysis modules are selected based upon the file type.

Next, the samples are scanned by several anti-virus (AV) engines. Results from any positive matches are stored in the database. Having multiple AV engines is crucial because it provides better coverage since no single AV engine is able to detect every known piece of malware. Due to the reactive nature of AV engines, malware that is submitted to FARM soon after discovery often will not have any positive matches from the AV scanners. An option is available to re-scan the sample at a later date when signatures may have become available.

For Windows PE files, the next module is a PE header parser. This module gathers information such as the imported DLL functions, the sections and their sizes, the entry point for code execution, etc. This information is useful for grouping malware that has been packed or obfuscated by similar tools.

A third-party tool, PEiD, provides the next two analysis modules: packer detection and crypto detection [8]. PEiD performs signature-based detection of packer algorithms and compilers. It also includes a plug-in, Kanal, that does signature-based detection of cryptographic algorithms. Since PEiD is only available with a GUI that runs on the Windows platform, it was necessary to script interaction with the GUI in order to automate the analysis. This is done in a Windows virtual machine. To script the tool, we chose to use a GUI scripting framework called AutoIt [9].

The behavioral analysis is one of the most important modules in FARM. The behavioral analysis module gathers information about the malware when it is executed. Currently, behavioral analysis is performed on Windows PE executables, but it is desirable to analyze many other formats (i.e. Microsoft Office documents, PDFs, JavaScript, ELF executables). The behavioral analysis relies on two other tools: TRUMAN, a sandbox environment developed by Joe Stewart [10], and ISLAND, which was developed at Sandia and is described below.

The Isolated Security Laboratory for Attacks and Network Defense (ISLAND) test bed is a framework for automating network and malware experiments. It is based on RADICL, a tool developed at the University of Idaho [11], [12]. ISLAND allows a researcher to rapidly configure and deploy diverse network topologies and operating systems on real hardware. In many other malware test beds, virtualization is used to set up and run experiments of this nature. ISLAND avoids

virtualization since often malware samples behave differently when they detect virtualization [13].

Operating system images are stored on a central server and deployed to the ISLAND testbed machines, where they are then modified to include any unique configuration required by the experiment. Part of this process allows outside files (i.e. malware samples) to be inserted into the file system. ISLAND allows experiment creation and operation to be fully automated using experiment templates and batch jobs.

When a new sample arrives in FARM, the behavioral analysis module creates an ISLAND job including the malware sample and a template to setup the TRUMAN environment. Once in the ISLAND system, the job is run immediately if resources are available or it is placed in a queue to be run when resources become available. Each sample is executed in the TRUMAN sandbox environment for a fixed period of time. Fig. 2 illustrates the TRUMAN environment.

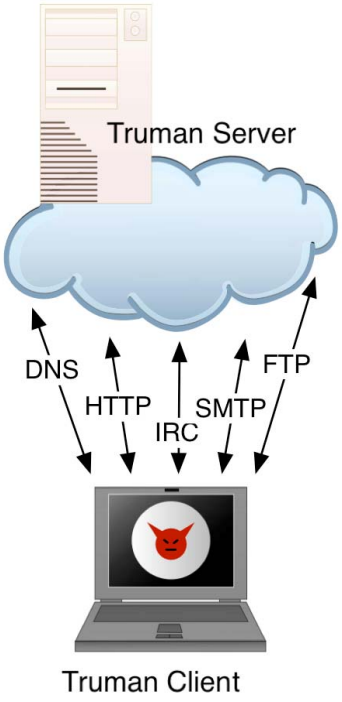

Fig. 2 - FARM Behavioral Analysis

The TRUMAN environment includes a client where the malware is executed as well as a server that runs several server applications including DNS, HTTP, IRC, FTP, and SMTP. These server applications are known as "faux servers," and their purpose is to coax the malware to interact with the server and divulge as much information about itself as possible. For instance, Trojan downloaders often make an HTTP request to download the next stage for a malware infection. During the FARM behavioral analysis, such a Trojan downloader running on the TRUMAN client would cause a DNS request to be generated requesting the IP address for a domain. The faux DNS server responds with its own IP address. This would be followed by an HTTP request from the malware for the specific URL encoded in the downloader and some generic response.

The system gathers network traffic during execution of the malware. Following the execution, a disk image is acquired and analyzed to look for modifications to the file system and registry (if applicable). Additionally, the network traffic is analyzed for interesting queries and other pertinent information. All of this information is then packaged and sent back to FARM where it is processed and stored in the database. In the example of the Trojan downloader mentioned previously, the DNS and HTTP queries would be part of the information stored in the FARM database.

Another key analysis module in FARM is the automated unpacker. Most malware utilizes some form of packing and code obfuscation to avoid anti-virus detection and hinder reverse engineering. In this module, the malware is again deployed to an ISLAND experiment where it is run with Saffron [14], an automated unpacker from Offensive Computing, LLC. Saffron uses dynamic instrumentation to trace and monitor execution flow at runtime. By tagging all process memory and watching for execution, Saffron can determine when the unpacking algorithm has finished and the original entry point of the malware is reached. Once this occurs, the relevant process memory containing the unpacked code is dumped. Once unpacked, the sample is resubmitted to FARM and processed again.

Results from each analysis module are kept in the FARM database and can be searched from the web frontend. This allows analysts to track trends in malware development and find similarities between malware strains. Historical information is also retained when subsequent scans are performed.

THE FARM ADVANTAGE

The most important contribution of FARM is that it brings an automated analysis system to analysts via a simple web front-end. Without FARM, an analyst familiar with the RE process who has an environment prepared for this purpose will most likely have to spend precious cycles manually running the malware through a suite of tools. This manual process can be time-consuming, causing costly delay in implementing remediation steps for a suspected victim. In contrast, using our current version of FARM, an analyst can submit a malware sample and, after a short delay, begin receiving malware-related signatures for network, file system, registry, etc to assist the analyst in determining remediation steps.

During the time in which FARM is examining suspicious binaries, the analyst can be performing other tasks important to the investigation. For example, he/she could be filing incident reports, notifying appropriate personnel, alerting central incident response teams, etc.

To the more experienced RE analysts doing in-depth RE, FARM can offer savings in time and effort when working samples are analyzed automatically. The database backend of FARM could also aid the analyst in searching for common attributes in other samples that others may have already analyzed in-depth.

Additionally, FARM can provide information to other users who may not have the technical skills or experience necessary to set up the environment and manually iterate through the RE process. This includes IT specialists and computer security professionals not versed in the RE process.

CASE STUDIES

Besides scanning suspicious files routinely found coming into our networks for known viruses, FARM has been useful in rapid incident response scenarios where responders needed to quickly determine whether a machine had been infected. This knowledge is necessary to determine the remediation response. During such times, many questions need to be answered. For example: *do we need to disconnect the host machine from the network? Do we need to launch a more in-depth forensics analysis into the attack vector used to compromise the host? How do we better protect our networks against similar threats in the future?*

FARM has also been used to aid in more rigorous RE analyses of particular malware samples where output from the automated system helps to determine the RE analyst's next steps and/or data from the RE analyst is submitted into FARM to further the RE process. Following are several typical scenarios collected from users of FARM over the past six months.

Scenario 1

In routine network traffic analysis, an IDS analyst notices that a host on the network downloaded a suspicious Windows executable following a series of web redirections. Questions began to emerge in the analyst's mind. *Is the file malicious? If so, did the host machine's anti-virus engine catch the executable before it was executed? What are its file system and/or network signatures?* The analyst obtained the executable submitted it to FARM. Within a few seconds, preliminary results showed that the executable was not detected by any of the anti-virus engines available in FARM except for one. The AV that did positively detect the malware classified the executable as `Infostealer.Banker.C`. In addition, FARM reported that the binary used the functions `SetWindowsHookExA` and `GetWindowTextA` from the `user32.dll` library. This is an indication that the executable was attempting to steal information from the user's windowed applications. While waiting for the behavioral results to be returned by FARM, the analyst shared the preliminary information about the executable with other analysts. He also temporarily blocked networking access from the host. Once available, the behavior analysis results from FARM showed that a file called `ntos.exe` was created in the `WINDOWS/system32/` directory and that a registry entry was created which started this executable on Windows start. The results also showed that the executable made DNS requests for `microsoft-yahoo.name`. The `ntos.exe` file was verified to exist on the user's hard drive and steps were taken to remove the offending files from the system. The analyst then searched the network logs for other machines that may have contacted the `microsoft-yahoo.name` domain.

Scenario 2

In a forensic analysis case, an analyst was presented with some suspicious executables whose filesystem timestamp indicated that they were created after confirmed malicious software was installed. The goal was to determine the functionality of the suspicious files. The analyst submitted the files to FARM and saw that no useful behavioral information was generated but that one was identified by FARM as a Windows console application as opposed to a GUI application. This indicated that one of the executables may have been a command line tool downloaded by the malicious software to allow the remote attacker to do further damage. The initial strings analysis indicated that the executables were obfuscated; however, the Saffron [14] generic unpacking module of FARM was able to generate memory dumps from the original executable. These binary dumps were automatically resubmitted to FARM, which then extracted legible strings indicating that the executable was used to steal passwords from memory. Though strings analysis after unpacking proved useful in this case, it is important to note that a malicious code writer can easily build in bogus strings to confuse the analyst. Thus, disassembly analysis is suggested as a confirmation of the automated unpacking results.

Scenario 3

An RE analyst was asked to analyze a piece of malicious code associated with a recent intrusion. As soon as the sample was submitted to FARM, the RE analyst was informed by the system that another RE analyst had already analyzed the binary with the same exact MD5 sum. This saved the RE analyst a tremendous amount of time.

Scenario 4

A security analyst has used FARM in time sensitive situations where he needed successful compromise indicators quickly. The indicators by FARM, especially the behavioral module, proved very useful in determining whether a piece of malicious software had been successful or not. In one instance a malicious email was received. The malicious attachment was run through FARM and not detected by any of the antivirus scanners, which prompted the analyst to prepare alternate measures to protect assets.

Scenario 5

In a deep dive analysis for a complex binary executable, an RE analyst manually unpacked an obfuscated executable after studying the pattern revealed by the packer's disassembly. To check his work and make sure that the unpacked malware executed properly, he submitted the resulting extracted executable to FARM to see if the behavioral signatures were the same as the original packed executable. Matching results proved that the executable had been correctly unpacked. The unpacked executable could then be reverse engineered more easily since the code could be altered to force execution along a particular branch of code in the malware.

FUTURE WORK

FARM was developed to be modular, so that as malware analysis tools are developed, they can be plugged into FARM as an analysis component. In the short term, we are planning on adding more analysis modules such as AV scanners and unpackers. This includes unpackers specific to a known

packing algorithm, as well as generic unpackers that attempt to unpack all obfuscated malware. We are also improving and adding features to the behavior analysis module to improve the amount of data gathered during dynamic analysis as well as expanding the number of file types that can be analyzed.

Currently, samples are submitted to FARM through the web interface. This is intended to allow analysts to submit samples as they come across them during normal network security operations. In order to increase the number of samples and analysis results in the FARM database we will incorporate automated collection using honeypots and honeyclients. Additionally, we are looking for existing large collections of malware to feed into the FARM system.

One of the original reasons for creating FARM was to provide a way to classify malware. It is clear that the number of unique samples of malware is increasing and will continue to do so in the foreseeable future. As has been discussed, it is unreasonable to expect that the security research community can provide sufficient RE resources for analysis of each new malware sample. The samples themselves possess so many different attributes that it is difficult to determine whether or not a sample is "interesting" and should therefore be analyzed further by an RE expert. We intend to use machine-learning algorithms to classify malware as interesting or non-interesting based on attributes that are stored in the FARM database. Additionally, we may be able to group malware together based on the attributes, determine common authorship, and track malware evolution.

CONCLUSIONS

We have presented a platform for automating malware analysis called FARM, the Forensic Analysis Repository for Malware. The principal goal for FARM is to increase the rate at which we can perform quality RE and malware analysis. FARM integrates open source and commercial analysis tools into an automated system in order to reduce the time spent by analysts in evaluating new malware samples. FARM assists analysts in two ways: first, FARM allows analysts to automate an entire series of RE steps previously performed manually; second, automated analysis more quickly points the analyst toward new or more interesting malware, focusing precious RE resources on the most significant threats. Additionally, FARM makes some RE tools and techniques accessible to security analysts who lack RE training, thereby broadening the base of personnel who can participate in malware analysis.

FARM's modular, closed-loop design allows additional analysis tools to be easily integrated into the FARM framework, as well as allowing re-introduction of prior analysis results on a particular piece of malware back into the system for deeper inspection.

We have used FARM to support computer security operations at Sandia for the last six months, and FARM has saved valuable analyst time, as we intended. Specifically, FARM has made the task of rapidly assessing a binary to determine if it is malicious much easier, and has also greatly reduced the time needed to determine whether an attack on a host was successful. FARM has also greatly expanded the number of analysts capable of doing rudimentary RE of malware. Use of FARM has greatly augmented Sandia's malware RE capabilities and improved our overall computer security posture.

ACKNOWLEDGMENTS

We thank Karl Anderson, Jim Hutchins, Randy McClelland-Bane, Tan Thai, Eric Thomas, and Tim Toole for their contributions and input to our work.

REFERENCES

[1] C. Willems, T. Holz, and F. Freiling: "Toward Automated Dynamic Malware Analysis Using CWSandbox," *IEEE Security & Privacy*, 5 (2), pp. 32-39, 2007.

[2] Norman Sandbox, http://www.norman.com/microsites/nsic/.

[3] J. Oberheide, "Detecting and Evading CWSandbox," http://jon.oberheide.org/blog/2008/01/15/detecting-and-evading-cwsandbox/, 15 January 2008.

[4] A. Vidstrom, "Evading the Norman SandBox Analyzer," http://www.ntsecurity.nu/onmymind/2007/2007-02-27.html, 27 February 2007.

[5] N. Albright, "Malware evading sandboxes using rootkits," http://www.disog.org/2007/09/blog-post.html, 3 January 2007.

[6] Threat Expert, http://www.threatexpert.com/.

[7] Virus Total, http://www.virustotal.com/.

[8] PeiD, http://www.peid.info/.

[9] AutoIt, http://www.autoitscript.com/autoit3/.

[10] J. Stewart, *TRUMAN*, http://www.secureworks.com/research/tools/truman.html .

[11] The University of Idaho RADICL - http://www2.cs.uidaho.edu/~radicl/.

[12] S. Caltagirone, P. Ortman, S. Melton, D. Manz, K. King, R. Blue, and P. Oman, "A Reconfigurable Attack-Defend Instructional Computing Laboratory," *The 2005 World Congress in Applied Computing*, Las Vegas, Nevada, June 2005.

[13] J. Rutkowska, "Red Pill... or how to detect VMM using (almost) one CPU instruction," http://www.invisiblethings.org/papers/redpill.html, April 2007.

[14] D. Quist, Valsmith, "Covert Debugging: Circumventing Software Armoring Techniques," BlackHat 2007, Las Vegas, Nevada, August 2007.