Virus Detection Based on Automated Behavior Analysis Using Support Vector Machines

Scott Norman
UMBC
mr.scott.norman@gmail.com

Jef Harkay
UMBC
incutonez@hotmail.com

December 13, 2011

Introduction and Purpose

Traditional virus detection has been handled by comparing the properties and contents of files with a database of known virus signatures. This is no longer as effective with the increased use of polymorphic [1] code and the open resources on the Internet. Variants are created quickly and frequently and the delay in pushing signature updates to customers has led to signature detection no longer being a sustainable direction in hardening a computer's defenses.

Include the previous research in this section based on the papers we've read.

This project is an exercise to learn more about analyzing malicious software and how machine learning algorithms can be used to find relationships in the behavior of that software as it runs on an operating system. Based on our research, we know it may be possible to do this with Support Vector Machines. As an exercise and comparison, we will also utilize a random classification algorithm and a k-nearest neighbor algorithm in our experiments and results.

Problem Definition

Based on the collection of features from a collection of virus infected executables and normal, healthy executables, is there a relationship between their interactions with the operating system? We believe that a virus can be distinguished from a normal executable by observing the relationships between the various reads and writes to different parts of the operating system. The Support Vector Machines machine learning algorithm should provide the most accurate results in comparison to a random selection algorithm and an implementatino of the k-nearest neighbors algorithm.

Materials and Methods

Several software components are needed in order to collect the required data and perform our analysis. The first component involves a sandbox environment to safely run and gather runtime data from virus and benign software applications. The data gathered from this environment required additional manipulation to transform it into the master data set that we could input into our various algorithms. Lastly, there were three algorithms that we chose to investigate for this project including a simple random classification algorithm, an implementation of the k-nearest neighbor algorithm, and our usage of the Support Vector Machines machine learning algorithm.

Sandbox Configuration

We utilized VirtualBox [2] as our virtual machine since it is free, open source software. The guest operating system image used was a copy of Windows XP Media Center Edition with Service Pack 2 installed. After installation and activation of the guest operating system, the virtual machine had its network capabilities disabled and no further security fixes or updates were applied. The host operating system consisted of a Linux and MacOS x hosts. The Linux hosts included Ubuntu 11.04 and 11.10. The version of MacOS X was 10.7.2 Lion.

In order to gather the events and interactions with the guest operating system we utilized an advanced monitoring tool for Windows called Process Monitor [3]. This program monitors everything that happens in an operating system including file interactions, directory interaction, thread and process behavior, network activity, and interactions with the registry. Process Monitor was running during the execution of each malicious and benign file used in our research.

The information gathered was dumped into a CSV file. This file was further analyzed to pull the features required for the various machine learning algorithms we employed.

Automated Sandbox Script

After gathering data from a set of 20 viruses and applications, it was determined that the work for gathering our statistics was too menial especially if we wanted a large data set and we wanted to replicate it quickly. To cut down on this the process was automated by utilizing a scripting API available with VirtualBox. This allowed a command line script to be written in Python to run and gather the required data from each of our benign and virus infected files.

```
VMstate ← current state
files ← read(benignDir)
start vm
for file in files do
    start Process Monitor
    execute benign file
    end benign file process
    create csv features
    end Process Monitor
end for
stop vm

files ← read(virusDir)
for file in files do
    start vm
    start Process Monitor
    execute virus file
    end virus file
    create csv features
    end Process Monitor
    stop vm
    revert to VMstate
end for
```

Algorithms

Write up a summary of the three algorithms chosen for comparison in the experiments: Random Selection, k-Nearest Neighbor, Support Vector Machines. Discuss the program implementations of each, how to use them, etc.

Experimental Data

In order to narrow the scope of the project, we chose to focus on two classifications for the files in our datasets: viruses and benign, normal applications. The data gathered from our sandbox environment was further analyzed to generate the statistical data required for each of the algorithms we are implementing and examining.

Viruses

Our definition of a virus is a malicious application that must be executed by a user in order to infect the operating system. A virus does not include other malicious software such as worms or trojans. The viruses we collected were all from a website titled VX Heavens [4]. This website is devoted to collecting malicious software of all categories. In total, VX Heavens has over 26,000 unique viruses. They separate their virus collection into several different types, ranging from operating system viruses to specific program viruses for powerpoint, acrobat reader, etc.

We chose to specifically examine the Win32 collection on VX Heavens. These Win32 files can be run by appending a ".exe" to the end of the downloaded file's name. When this file is executed, it does whatever that virus was intended to do. We downloaded a total of 113 files, but some of these files didn't execute therefore our total dataset consists of 95 viruses.

Benign

A benign application is simply one that is not infected by any piece of malware. Our benign file dataset consisted of executables from our newly installed Windows XP guest image. This copy of Windows XP had its network connection disabled and was therefore clean of any infections before our experiment. We had a total of 70 benign executables, ranging from system commands, Microsoft games, and other miscellaneous utilities.

Features

The data dumped from Process Monitor's logs was further analyzed to create the data set used in our experiments. There are a total of 26 unique events that we collected. These events are broken down into 3 categories: registry, file, and misc.

The registry events are anything related to editing the Windows registry – reading, writing, creating, etc. The file events are anything related to files on the hard disk – writes, read, checking if it exists, etc. The misc events deal with anything else – process creation, thread creation, directory checks, etc.

Table 1: Features

Registry	File	Misc
Key Writes	Create Map	Query Directory Existence
Key Reads	Create File	Process Create
Key Create	Read File	Process Start
Query Key Value	Query File Existence	Thread Create
Query Key Existence	Query Standard Information	
Set Value	Load Image File	
Enum Key	File System Control	
Enum Value	Query Attribute Tag	
Desired Access Maximum Allowed	Query Basic Information	
	Write to File	
	Set Allocation Information	
	Set Basic Information	
	Set Disposition Information	
	Set End of File Information	

Results

Include overall results from each of the algorithms, including some charts and graphs that give visual comparisons in their performance. Any adjustments or modifications to the data set after initial experiments should be listed here as well.

Random Classification

Tables and graphs for using the random classification approach.

k-Nearest Neighbor

Tables and graphs for using the random classification approach.

Support Vector Machines

Tables and graphs for using the random classification approach.

Conclusion

This will probably be a paragraph. We should have it focus on the hypothesis and the question as it compares to the data and results. Did SVM perform as well as we expected? How did it perform compared to our kNN and random implementations? What did we find from the experiment? Once we answer those types of questions, we may have some additional things to report based on the answers. We may explain some ways in which the project might be improved or introduce new questions that have come up as a result of the project. What will we be able to conclude and what areas can we not draw valid conclusions based on our data and results?

Sources

[1]: http://en.wikipedia.org/wiki/Polymorphic_code
[2]: https://www.virtualbox.org/
[3]: http://technet.microsoft.com/en-us/sysinternals/bb896645
[4]: http://vx.netlux.org/
[5]: "Automatic Analysis of Malware Behavior using Machine Learning," http://honeyblog.org/junkyard/paper/ma
TR-2009.pdf
[6]: "A Study of Detecting Computer Viruses in Real-Infected Files in the n-gram Representation with Machine Learning Methods," http://www.sec.in.tum.de/assets/staff/stibor/iea.aie.final.extended.pdf