1992

A Generic Virus Scanner in C++

Sandeep Kumar

Eugene H. Spafford
Purdue University, spaf@cs.purdue.edu

Report Number:
92-062

Kumar, Sandeep and Spafford, Eugene H., "A Generic Virus Scanner in C++" (1992). *Computer Science Technical Reports.* Paper 983.
http://docs.lib.purdue.edu/cstech/983

A GENERIC VIRUS SCANNER IN C++

Sandeep Kumar
Eugene H. Spafford

CSD-TR-92-062
September 17, 1992

A Generic Virus Scanner in C++ *

Technical Report CSD–TR–92–062

Sandeep Kumar   Eugene H. Spafford

The ***COAST*** Project
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907–1398
{kumar,spaf}@cs.purdue.edu

17 September 1992

Abstract

Computer viruses pose an increasing risk to computer data integrity. They cause loss of valuable data and cost an enormous amount in wasted effort in restoration/duplication of lost and damaged data. Each month many new viruses are reported. As the problem of viruses increases, we need tools to detect them and to eradicate them from our systems.

This paper describes a virus detection tool: a generic virus scanner in C++ with no inherent limitations on the file systems, file types, or host architectures that can be scanned. The tool is completely general and is structured in such a way that it can easily be augmented to recognize viruses across different system platforms with varied file types. The implementation defines an abstract C++ class, `VirInfo`, which encapsulates virus features common to all scannable viruses. Subclasses of this abstract class may be used to define viruses that infect different machines and operating systems. The generality of the mechanism allows it to be used for other forms of scanning as well.

*This paper appears in the Proceedings of the 8th Computer Security Applications Conference, IEEE Press, 1992.

1 Introduction

Computer viruses pose an increasing risk to computer data integrity. They cause loss of valuable data and cost an enormous amount in wasted effort to restore or recreate damaged or destroyed data. Each month new viruses are reported (cf. issues of ***The Virus Bulletin*** and ***Virus News and Reviews***). As the number of viruses increases, we need tools to detect them and eradicate them from our systems. While the problem of detecting, without error, all viruses automatically is intractable[4], it is certainly feasible to detect simple, known viruses.

In this paper we describe a tool – a generic virus scanner in C++ – that can run in a high integrity virtual memory environment and scan for viruses in files. These files can be available either on the tool's host machine itself, available through a locally mounted file system, or visible to it through an appropriate network mount. For example, the testing machine could be a UNIX workstation accessing a DOS filesystem through its floppy disk interface or remote-mounted on the workstation through a network (e.g., Novell or PC-NFS). The rapid proliferation and use of network software in the PC community has already created a need for such interfaces whereby PC mounted file systems and file servers may be accessible to more powerful workstations on the same local area network.

One common definition of a virus is as a segment of machine code that installs a (possibly evolved) copy of itself into one or more larger "host" programs[4].[1] When the program is executed, the code is activated and enables further spread of the virus, or destruction of data, or both. The principal cause of this problem is the almost nonexistent controls in most PC systems that allows user programs to potentially gain complete control of the system. This allows virus code to perform any operation, and to change any code or data.

Looking for viruses is not a simple matter of looking for extraneous code, because it is not always obvious what is extraneous. Recent "stealth viruses" make even this procedure difficult by ensuring that the original contents of an infected file are returned when its contents are requested as data for examination.[2] It is more reliable

[1] Other definitions may be found in the collections [7] and [10].

[2] See [8] for a good description of how stealth viruses operate.

to test for infected files by using a system that partitions its processes into distinct address spaces by a virtual memory translation. This way we can avoid the effects of stealth and other memory resident viruses on the scanning procedure. Better still would be the use of a scanner on a completely different architecture — one that cannot support the execution or spread of the searched-for viruses. In such an environment, when a virus scanner running as a user program requests bytes from a file for examination, it is assured of the integrity of the bytes from influence by other user programs; in no case can an ordinary user process modify the interrupt vectors of devices or traps leading to system calls.

If one is testing for viruses on a diskette using a machine that runs processes in their own virtual memory space, one can be reasonably sure about the integrity of the memory. Similarly, boot diskettes can be just as easily tested because we do not attempt to boot from the diskettes and, in the process, compromise the integrity of the testing machine in any way. In short, by testing on a machine with processes running in their own virtual address space we ensure, with very high confidence, the integrity of the machine on which we are doing the testing. This is similar to doing a high integrity boot of a PC before scanning for viruses on it. Furthermore, if we are able to run our detector in a completely different environment from the one containing the potential viruses, those viruses cannot infect or interfere with our detector.

2 How viruses are detected

In this section we review current methods of virus detection and end with the conclusion that detection by signature scanning still remains the most simple, economical and commonly used tool for virus detection.

2.1 Virus monitors/detection by behavioral abnormality

In this approach to virus detection, the machine is booted from uninfected files and a virus monitor is installed that monitors various activities of the machine while in day-to-day use. The program monitors known methods of virus activity including attempts to in-

DOCKET
ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase®
Smarter legal research.