DAZUKO: AN OPEN SOLUTION TO FACILITATE 'ON-ACCESS' SCANNING

John Ogness
H+BEDV Datentechnik GmbH, Lindauer Str. 21,
D-88069 Tettnang, Germany

Tel +49 7542 5000 • Fax +49 7542 5210 • Email jogness@antivir.de

ABSTRACT

One of the most fundamental forms of virus protection is at the file access level. By scanning files as they are opened or executed, malicious code can be blocked before having an opportunity to cause damage. However, with the constant evolution and availability of various operating systems, there is a continual redundant effort by anti-virus organizations to implement file access monitoring. This results in variable performance and a lack of support for certain platforms. This paper presents an open source project, Dazuko, which provides a standard interface for handling file access control. The project aims at developing the Dazuko module to work with many different operating systems while maintaining a common interface. By providing the anti-virus community with an open file access control standard, a broad range of supported systems with reliable performance can be established.

1. INTRODUCTION

In the ongoing fight against malicious code, on-access technology has become a powerful ally. With the ability to detect malicious code before the operating system has an opportunity to utilize it, users and their systems remain protected behind a real-time shield.

However, implementing on-access technology involves a close relationship with the operating system. An on-access scanner must have the ability to intercept low-level file access events before the operating system itself begins to work with the file. This intimate relationship may involve interfacing directly with the operating system or possibly modifying it, allowing a virus scanner to effectively carry out its work.

With the many varying flavours of operating systems available, the task of implementing an on-access interface becomes enormous. These varying systems not only take advantage of completely different architectures [1], but they also have different licensing procedures. Dealing with the many companies, interfaces, and platforms becomes a major issue. However, regardless of the difficulty, these steps must be taken in order to offer users what they need: a safe and virus-free computing environment.

Unfortunately all the various anti-virus organizations must independently develop the on-access capability themselves. This enormous effort is reproduced over and over again. Each organization must establish relationships and develop, debug, and test modules, which must flawlessly operate so as not to jeopardize the entire system.

These varying modules from varying organizations also often do not work alongside one another – which is understandable since they were developed independently. This means that a user is often unable to install multiple on-access anti-virus packages at the same time. Rather than anti-virus organizations offering users options for security, the user is forced to choose between packages that completely differ from one another in capability and reliability, and cannot coexist.

What is needed is a standard anti-virus interface available from the operating system itself. With such a standard implemented across all major operating systems, anti-virus organizations could more easily provide effective software while focusing on virus detection, rather than interfacing with an operating system. The user would no longer be burdened with choosing a package based on support, but rather on anti-virus capability and features.

2. AN OPEN SOLUTION

Dazuko, pronounced 'dah-tsu-ko', is the name of an open source project [2], which began as an effort to address the current discontinuity and frustration facing on-access scanner developers and users. Although not capable of doing any type of virus scanning itself, *Dazuko* provides a simple interface for other third party applications (virus scanners) to control file access. It can be viewed as an operating system 'plug-in' that provides an interface for an on-access anti-virus mechanism. By supporting many different operating systems, a common and familiar interface is available to developers on multiple platforms.

Dazuko is available under a BSD license [3], meaning that organizations are free to use and modify the source code as long as they include the original copyright and licence conditions. Changes to the code are not required to be recommitted to the project and integrating *Dazuko* into existing software does not put additional restrictions or requirements on that software.

The licence is flexible enough that both closed and open source organizations can safely integrate *Dazuko* into their code base. The openness of the project and its immediate value to organizations helps to promote a positive and cooperative development environment.

2.1 Concept

Dazuko works directly with the operating system kernel to intercept file-accessing system calls. In order for an application to utilize *Dazuko*, it must first register itself. By registering, the application communicates to *Dazuko* that it is prepared to execute file access control. (It is important to mention that the registered application runs in userspace and not as a kernel process.)

Once a file access event occurs (such as the user opening a file), *Dazuko* intercepts this event and notifies registered applications. Information about the event such as file name, type of access, various access flags, etc., is provided. Using this information, the registered application must then decide if the file access is to be granted or denied. For the case of anti-virus software, the application will scan the file associated with the access event and allow access if the file is virus-free.

While the registered application is determining whether the access should be allowed, *Dazuko* patiently waits for a response. Depending on the decision of the registered application, *Dazuko* will have the operating system either continue to process the file access normally or return an error, thus preventing the file access from ever occurring. Since *Dazuko* integrates into the operating system, its existence is completely transparent to running user applications. When file access is denied, the user application receives error codes from the operating system itself, and not from any extra module. This is important because it allows existing software to run reliably without the need for special adaptations.

In order to support multi-threaded operating systems, *Dazuko* is capable of working with multiple registered processes (or threads) from the same application. These processes can work together to share the work of file access control. When a file access event occurs, *Dazuko* looks for the first available registered process of the group and assigns the file access event to that process. Multiple processes signify that they are working together by registering under the same group name. This concept can be seen in Figure 1. The result is a multi-process on-access scanner that takes advantage of multi-processing operating systems, reducing the noticeable effect of additionally scanning files as they are accessed.

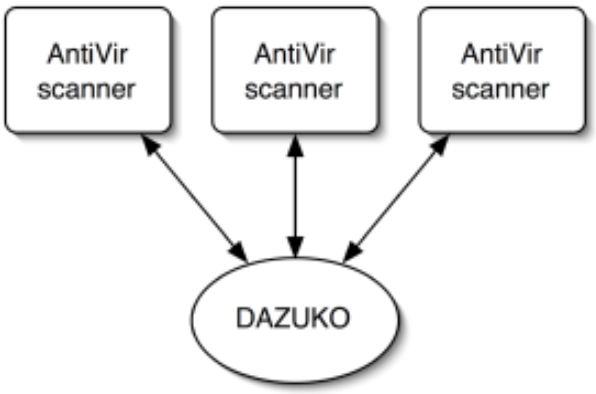

Figure 1. Sharing the work.

Three *AntiVir* scanner processes register under the same group name, thus sharing the responsibility of file access control. *Dazuko* assigns a file access event to the first available process in the group.

Dazuko also supports cascading, which is a feature allowing different applications to run together at the same time. This is similar in concept to the multi-process support mentioned previously, except that file access control is queued rather than shared. The applications distinguish themselves from one another by registering under different group names. With each file access event, one registered process from each group is given a chance to determine if an access should be allowed or not. If any one of the processes determines that access should be denied, then the access is denied. However, regardless of which process denies access first, all registered groups are given an opportunity to investigate the file access event for themselves. Figure 2 shows an example of two different applications utilizing *Dazuko* simultaneously.

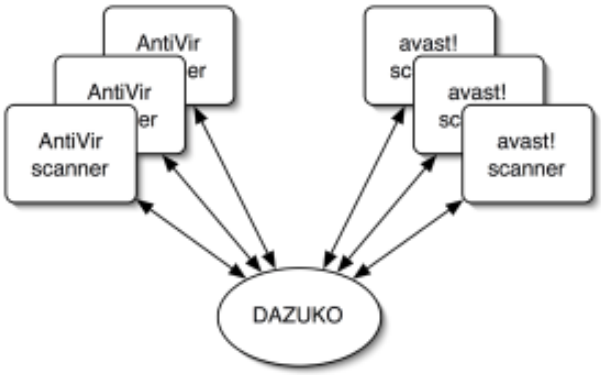

Figure 2. Queuing the work.

Two different applications (each with three processes) are running simultaneously with *Dazuko*. The *AntiVir* processes have registered using a particular group name and the *avast!* processes have registered using a different group name. *Dazuko* assigns a file access event to the first available process in the *AntiVir* group and then the same file access event to the first available process in the *avast!* group.

2.2 Interface

The *Dazuko* interface is quite simple. It has functions to register, set parameters, request an access event, return an

access event, and unregister. Regardless of which operating system (or language) an application is written in, the *Dazuko* interface remains the same. This allows easy cross-platform development from third party developers. Figure 3 shows a surprisingly simple example of an on-access scanner implemented in C. Details about the interface as well as example programs are available from the *Dazuko* project website [2].

```
int main()
{
    struct access_t access;

    dazukoRegister("myScanner");

    dazukoSetAccessMask(ON_OPEN | ON_CLOSE | ON_EXEC);
    dazukoAddIncludePath("/");

    while (dazukoGetAccess(&access) == 0)
    {
        access.deny = do_scan_file(access.filename);
        dazukoReturnAccess(&access);
    }
    dazukoUnregister();

    return 0;
}
```

Figure 3. A simple on-access scanner.

Note: do_scan_file() is implemented elsewhere.

2.3 Layers

In order to keep the *Dazuko* file access control 'plug-in' cross-platform, three different layers have been developed. The first layer is the platform-dependent layer. Here a set of functions are implemented in a platform-specific manner. This is the layer that does all the real 'work' of interacting with the operating system. It is also the layer which must be completely re-written for each supported operating system.

The second layer is the functionality layer. This layer implements the 'brains' of *Dazuko*, utilizing the platform-dependent layer's interface to do get the work done. All decision making and handling is done in this layer. The only changes to the functionality layer represent bug fixes or new features.

The third layer is the visible layer. Here the public interface for *Dazuko* is made available to applications. This layer's only responsibility is to provide a front-end through which the functionality layer can exchange information with the application. Changing this layer is done only if absolutely necessary since it could lead to incompatibility with existing applications that rely on *Dazuko*.

All three layers and how they communicate with each other can be seen in Figure 4.

In order to easily maintain portability with a constant interface, *Dazuko* is composed of three layers.

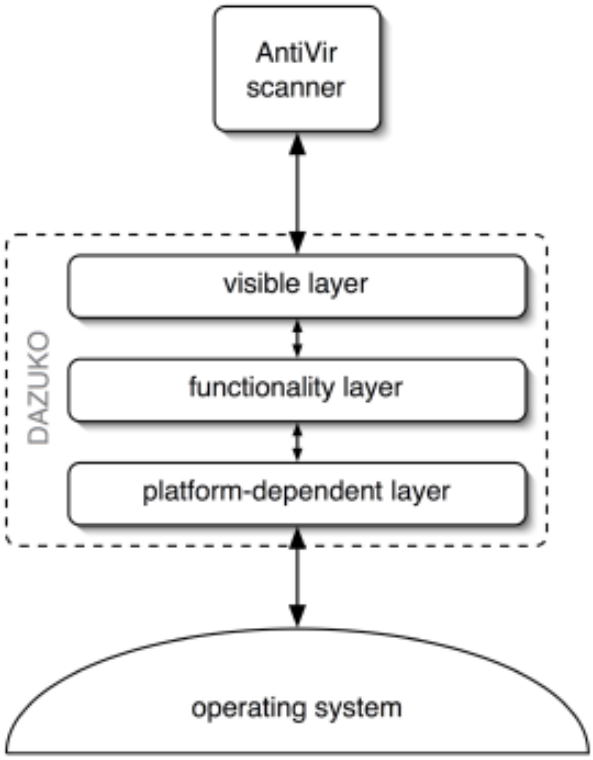

Figure 4. Layers in Dazuko.

There are three main advantages to having *Dazuko* implemented in these three layers. First, by keeping the functionality layer free of platform-dependent code, feature implementation and debugging is made easier. This allows the 'pure' functionality to be clearly seen and enhanced. Modifications made to this layer provide improvements uniformly across all platforms.

Another advantage is the concrete interface defined for the platform-dependent layer. Porting *Dazuko* to other platforms becomes a fairly straightforward process of implementing many basic platform-specific functions. Building *Dazuko* on another platform simply involves swapping out platform-dependent layers.

The third main advantage is the abstraction provided by the layering model, which allows the visible layer to maintain a constant application interface across platforms. This is especially important for organizations to effortlessly develop cross-platform on-access applications. The interface always remains the same, with the most difficult tasks of on-access implementation (licensing, kernel interfaces, special relations and skills) already completed by the *Dazuko* project.

3. RESULTS

Since its release in February of 2002 [4], Dazuko has already achieved a considerable amount of recognition

[5][6][7][8][9][10]. Many organizations have begun to investigate *Dazuko*'s capabilities and realize that the project should be taken seriously and has a great deal to offer [11][12].

Support for both *Linux* and *FreeBSD* kernels has already been implemented. This was first made possible after the development of the three-layer abstraction model. With the layer implementation complete, *Dazuko* can quickly move forward to begin supporting other operating systems.

Dazuko has received contributions from various people in the open source world. In fact, the release of version 1.1.2 of *Dazuko* was based entirely on outside contributions, including optimizations, devfs support, and patches for *Red Hat Linux* functionality [13]. Tapping into the global community has added extra development skills and experience, demonstrating one of the advantages of the *Dazuko* project's openness.

The visible layer is not only available in C, but has been ported to Java as well [14]. This has expanded the developer audience for *Dazuko*-based applications and extended the cross-platform capabilities of *Dazuko* to the application level.

Dazuko has succeeded in bringing various anti-virus organizations together by providing them with a common basis for their software. Although this cooperation does not reveal any formal alliances, it does unite organizations to producing a single, robust interface for anti-virus protection. By contributing to its development (either through usage or through code), these organizations are helping to promote a better, more secure, and more reliable foundation.

In a recent *Virus Bulletin* test of *GNU/Linux* scanners [15], Matt Ham mentioned that on-access scanning was a feature that was lacking in many products. Those organizations that were based on *Dazuko* provided uniform installation and capability (with respect to core on-access technology). The encouraging words throughout the article clearly portray a need for a standard solution and suggests *Dazuko* as one such possibility.

4. FUTURE GOALS

Although *Dazuko* has already proved its usefulness on several server platforms, it must continue to expand. One of the main upcoming tasks of the *Dazuko* project is to begin porting to mainstream desktop operating systems. These include the various versions of *Microsoft Windows* and MacOS.

However, before moving into mainstream desktop environments, *Dazuko* must first offer better documentation and installation tools. The combined expertise of various organizations may provide information that will make it easier for users to install and use *Dazuko* with their applications. This will be a collaborative effort since it is the desire of every party that *Dazuko* is easy to use.

After porting to mainstream desktop operating systems, *Dazuko* will be presented with an opportunity to greatly expand its number of users. This will indeed put *Dazuko*'s abilities to the test, as many thousands of users will become dependent on *Dazuko* for their anti-virus protection. It is important that these steps are taken carefully and in a secure and stable direction.

It is also important that *Dazuko* expands its partners, portability, and availability. Such examples include working with various operating distributions, such as *Red Hat* [16] and *FreeBSD* [17], to provide pre-packaged *Dazuko* software. Existing distribution packages, such as those from SuSE [18] and Debian [19] must be kept current with the latest software and documentation. The *Dazuko* project should also form relationships with security projects, such as RSBAC for Linux [20], which have a great deal of experience developing secure kernel systems. *Dazuko* can also build other alliances, such as the VTrace [21] or FAM [22] projects, which have implemented very similar functionality to Dazuko except with alternate objectives. These various code bases could be integrated, representing a powerful collaborative alliance of development experiences.

In an effort to become a standard for third party file access control, the *Dazuko* project must acknowledge existing standards. It is vital that *Dazuko*, in becoming a standard, remain compliant with existing file access control protocols and interfaces. The OPENXDSM Open Group Standard is one such example [23]. Although it is clear that some of these standards support more functionality than is necessary (or desired), *Dazuko* has a responsibility to acknowledge these standards and move in a direction of compliance.

Finally, *Dazuko* needs an improved security model for determining trusted applications for registration. The version available at the time of this writing relies on root (administrator) privileges of the registering application. These applications are then automatically trusted. A more robust method must be developed, which allows true application authentication during registration. Through cooperation with partners, this difficult task can be resolved by incorporating the experience of many organizations.

5. CONCLUSION

Although *Dazuko* is still very young, it has already begun to prove its value. Several anti-virus organizations have used *Dazuko* for their *GNU/Linux* on-access scanning solutions and *Dazuko* has been acknowledged from many various sources. With its continued recognition and adoption, it is well on its way to becoming a standard in the anti-virus industry. By working together, the community is helping to build a strong, standards-based foundation allowing

anti-virus organizations to focus on what they do best … detecting viruses.

As *Dazuko* becomes an accepted standard, its installation and use will become easier, it will become more reliable, and it will improve in performance. This translates to better and uniform support by anti-virus organizations on multiple platforms, common installation procedures, and freedom for the user to choose which anti-virus software should be installed.

6. REFERENCES

[1] Hayes, Bill, September 2002, http://www.securityfocus.com/infocus/1622.

[2] Dazuko Project, http://www.dazuko.org/.

[3] Dazuko Licence, http://www.dazuko.org/LICENSE.txt.

[4] Ogness, John, 'Dazuko is free software!', http://savannah.nongnu.org/forum/forum.php?forum_id=414, February 2002.

[5] Annuscheit, Rainer, 'Standard-Interface für externen Dateizugriff unter Linux', *IT SecCity*, November 2002. http://www.itseccity.com/content/dailynews/021118_dailynews_text.html.

[6] 'Dateizugriffskontrolle fuer Linux', *Linux Magazin*, January 2003, p.11.

[7] 'H+BEDV bietet Standard-Interface für externe Dateizugriffe an', *entwickler.com*, November 2002, http://entwickler.com/itr/news/show.php3?nodeid=82&id=8133.

[8] Lepish, Martin, 'Avast! 4.0 pre Linux', *Virusy.sk*, May 2003, http://www.virusy.sk/clanok.ltc?ID=366.

[9] Muench, Martin, 'Safer Tux', *Linux Enterprise*, December 2002, pp. 46–47.

[10] Schroeper, Joerg, 'Dateizugriff kontrollieren', *Unix Open*, January 2003, p.13.

[11] 'avast! 4 for Linux', *avast!*, June 2003, http://www.avast.com/i_idt_172.html.

[12] *Clam Antivirus: User Manual*, October 2002, http://clamav.elektrapro.com/doc/clamdoc.pdf.

[13] Ogness, John, Dazuko 1.1.2 released, January 2003. http://mail.gnu.org/archive/html/dazuko-devel/2003-01/msg00000.html.

[14] Ogness, John, Dazuko 1.2.0 released, May 2003. http://mail.gnu.org/archive/html/dazuko-devel/2003-05/msg00000.html.

[15] Ham, Matt, *Virus Bulletin*, May 2003, pp. 18–23.

[16] Red Hat, http://www.redhat.com/.

[17] FreeBSD, http://www.freebsd.org/.

[18] km_antivir, SuSE, http://www.suse.de/us/private/products/suse_linux/i386/packages_professional/km_antivir.html.

[19] dazuko-source, Debian, http://packages.debian.org/unstable/utils/dazuko-source.html.

[20] RSBAC, http://www.rsbac.org/.

[21] Lorch, Jacob and Smith, Alan, 'The VTrace Tool: Building a System Tracer for Windows NT and Windows 2000', *MSDN Magazine*, October 2000.

[22] IMon, http://oss.sgi.com/projects/fam/.

[23] OPENXDSM Open Group Standard, http://openxdsm.sourceforge.net/.