The Next Wave

The National Security Agency's Review of Emerging Technologies

The Next Wave

NSA's Review of Emerging Technologies

Letter from the Editor

Open source software (OSS) is a growing part of the software market. Although much OSS is low cost, even free, a great deal of its growing popularity is based not on cost but on the ability to see and manipulate the software internals. This ability enables great flexibility, allowing programmers to easily build upon or extend software to meet specific mission needs. Increasing interest in OSS is not limited to commercial markets; the US Department of Defense (DoD) is looking for ways to harness the community development model within internal projects. DoD has even created its own community software site, milforge.mil, to mimic the public open source community. In this issue of *The Next Wave* (TNW), we have a series of articles about important projects that were enabled by open source software.

The article "Raising the Bar in Operating System Security: SELinux and OpenSolaris FMAC," written by Steve Smalley, presents an update on Security-Enhanced Linux and introduces a project to bring similar security enhancements to Sun's OpenSolaris operating system. The next article, "Providing a Secure Foundation for Applications with the Certifiable Linux Integration Platform", by Todd Paisley, Brandon Whalen, and Stephen Lawrence introduces an NSA project to enable quicker and cheaper deployment of secure system solutions to meet mission needs. The article, "Cryptographic Binding of Metadata" by Calvin Mason, calls attention to the importance of trustworthy metadata and presents a summary of two NSA projects to protect metadata. Finally our focus article, "Open Source Software—A Growing Trend," by Russ Sutcliffe and Ray Young, explains the impact open source is having on the global software market.

Tux, the creation of Larry Ewing (lewing@isc.tamu.edu), has represented Linux since 1994. The original Tux logo was created with GIMP—the GNU Image Manipulation Program. Custom versions of Tux will serve as the guide through this open source edition of *The Next Wave*.

The Linux penguin and GIMP are two examples of open source products available free to the public for use and modification.

CONTENTS

Taking the Open Source Road

The direction is clear. Open-source software is paving a path to the information-centric future envisioned by the U.S. Department of Defense (DoD).

In an address to Department personnel, the Pentagon's deputy chief information officer (DCIO), David M. Wennergren, explained the reason for taking this new direction.

"In today's world we have to share information with people we never even dreamed of, using tools and means we never thought of before, [in] non-traditional ways with non-traditional organizations. And that's the power of the information world."

Open-source software (OSS) has proven to be a powerful tool for distributing, improving, and extending new programs. OSS licensing is gaining acceptance by industry and governments worldwide. It is the power of openness derived from collaboration and innovation that DoD intends to harness.

For nearly two decades, NSA's Research Directorate (RD) has been exploring the role OSS can play in helping the DoD achieve its missions. Efforts by RD researchers led to the public release in December 2000 of security-enhanced (SE) Linux. Since then, developers from HP, Hitachi Software, IBM, NEC, Red Hat, and other commercial as well as private institutions have contributed to extending the program's features and maturing its functionality. (For an in-depth look at SELinux, see "Raising the Bar in Operating System Security" in this issue of *The Next Wave*.)

More recently, DoD developers contributed over a million lines of code to the open-source (OS) community, helping expand the federal OS highway. One million lines is about the size of the Corporate Management Information System (CMIS), developed internally by the U.S. Defense Information Systems Agency (DISA). DISA spent almost a decade building the web-based federal workforce management, workflow, and administrative software suite for use by more than 16,000 military personnel.

When other government departments wanted to adopt the CMIS suite of more than 50 applications, DISA's director of manpower, personnel, and security, Jack Penkoske, asked, "Why not let them?" And if sharing CMIS with other federal agencies was a good idea, why not include academia, industry, and the entire OS community?

DISA did just that. In March 2009, the Pentagon's information technology unit announced a cooperative research and development agreement (CRADA) with the Open Source Software Institute to make an OS version of CMIS available to other federal agencies, academia, non-profit organizations, and industry to reuse and improve. DISA used Adobe Cold Fusion and Microsoft SQL Server 2005 to build OSCMIS. The department's effort was recognized by Government Computer News (GCN) as one of the "11 great government IT projects" for 2009.

Other federal OS projects have also come on line recently or are in the works. In June 2009, the U.S. Air Force released a free and open-source version of FalconView, a personal computer-based mapping application developed for the DoD by Georgia Tech Research Institute. Even the White House has entered the OSS arena. The popular WhiteHouse.gov web site switched from a proprietary content management system to open-source Druple in October 2009. Citing the potential for everyday citizens contributing to the evolution of the web site, White House media director Macon Phillips said, "We're looking forward to getting the benefit of their energy and innovation."

The open-source road at times has been a bumpy one, not just for the DoD, as IT planners attempt to steer a course straddling the benefits of openness with the demands of security. But Wennergren, who also serves as Deputy Assistant Secretary of Defense for Information Management Technology, argues that this is the wrong way to look at the challenge. He proposes a different model in which openness and security are goals that can reinforce each other.

The DoD has officially endorsed the use of OSS, stating as much in guidelines published in 2003. A memorandum released 16 October 2009 by the CIO team provides "clarifying guidance" intended to overcome "misconceptions and misinterpretations of the existing laws, policies, and regulations that deal with software and apply to OSS." DCIO Wennergren states in the memo that these *misconceptions* and *misinterpretations* have hampered DoD efforts to effectively develop and use OSS.

OSS has already been adopted by several DoD agencies, with its extensive use cited as early as 2003. In a study prepared for the DoD, "Use of Free and Open-Source Software (FOSS) in the U.S. Department of Defense," MITRE Corporation found that "FOSS software plays a far more critical role in the DoD than has been generally recognized." Now, the 2009 memorandum, which supersedes DoD guidelines of seven years ago, opens the door to even greater proliferation of OSS products in the future.

One force accelerating the DoD's adoption of OSS may come from the memo's affirmation that "in almost all cases, OSS meets the definition of 'commercial computer software'," and needs to be

treated as such. Classifying OSS as a commercial item grants it special contractual consideration. U.S. law (10 USC 2377) calls for government procurement to favor commercial supplies and services. The law requires heads of agencies to ensure that procurement officials are trained in the acquisition of commercial products. Everyone involved in the procurement process must, by law, make every effort to acquire commercial products and require contractors to incorporate commercial products, when possible. Additionally, specifications for contract requirements should be stated in terms that encourage bidders to supply commercial products, and procurement policies, practices, and procedures are to be written or revised to reduce impediments to their acquisition.

Policies set out in Federal Acquisition Regulation (FAR) 10.001 and the DoD supplement, DFARS 252.227-7014, specifically require executive agencies to include OSS when conducting market research for software procurements. This obligation places OSS products on equal footing with all other commercial products.

Still, misconceptions and misinterpretations stand in the way of government agencies adopting OSS, according to DCIO Wennergren. Ambiguous terminology associated with OSS has led, in part, to some of these misconceptions. Common usage of the word *free* has contributed to the mistaken understanding that freeware does not qualify as a commercial product. The reference to "free" in FOSS or "libre" in FLOSS (free/libre open-source software) applies to *free* or *libre* access to use, modify, and extend a product's underlying code, and not to the cost of the software. Both the Free Software Definition (FSD) and the Open Source Definition (OSD), the two main documents governing OSS development and use, support this interpretation.

By U.S. Government standards open-source software is not necessarily *free* software, even when it is available free of charge. When Congress enacted the No Electronic Theft Act (NET) of 1997 (HR 2265) to criminalize copyright infringement done "willfully and for purposes of commercial advantage or private financial gain," the definition for "financial gain" was added to Section 101 of U.S. copyright law. The U.S. Patent Office now considers the "receipt, or expectation of receipt, of anything of value, including the receipt of other copyrighted

works" financial gain. By applying this definition to OSS, simply improving code—something encouraged by the open-source community and expressed in the OSD and the FSD—is a form of financial gain, making the software *commercial*.

More obvious efforts to commercialize OSS are gaining momentum, as well. Research firm IDC forecasts a 22.4 percent compound annual growth rate (CAGR) for OSS revenue worldwide over the next five years. That gain would push revenue from OSS past $8 billion by 2013. In addition to established OSS vendors like Red Hat and Novell, service-based IT businesses such as IBM and Oracle have been tapping the open-source market by offering OSS support. Google's OSS strategy has been to shift emphasis away from the code in favor of transparency and application programming interfaces (APIs). Even Microsoft is promoting efforts to support interoperability between the company's proprietary line of products and OSS.

In addition to assuring DoD agencies *it's OK* to use OSS and requiring them to include OSS solutions in the procurement process, DCIO Wennergren listed seven "positive aspects" of OSS that DoD departments should consider when conducting market research for selecting software.

1. *Continuous and broad peer-review:* Publicly available source code is open to greater scrutiny than a software development team can provide. Continuous and broad peer review promotes improved software reliability and security.

2. *Unrestricted ability to modify software source code:* Agencies can initiate code changes rapidly in response to changing situations. Unrestricted access to source code makes it easier to enhance missions and respond quickly to emerging threats.

3. *Reduced barriers to vendor entry and exit:* OSS can be operated and maintained by multiple vendors. Eliminating proprietary restrictions reduces reliance on a particular software developer or vendor.

4. *Flexible deployment:* Open-source licenses do not restrict how software is used or by whom. This network-centric licensing model enables rapid provisioning of both known and unanticipated users to meet changing mission and user requirements.

5. *Reduced cost for licenses:* Because OSS typically is not licensed on a per-seat basis, costs are lower for large distributions, and users can be added at any time.

6. *Reduced cost of ownership:* All users of OSS share the responsibility for its maintenance. The overall cost of software ownership is therefore reduced for all parties.

7. *Rapid prototyping and experimentation:* OSS is particularly suitable for software development. The ability to "test drive" the software with minimal costs and administrative delays provides added benefits.

How well commercial software measures up against these seven criteria should be calculated in the evaluation of any software acquisition, whether it is proprietary or open source. Although these positive aspects of OSS are deemed relevant for a market survey, the CIO memo cautions they shouldn't be interpreted as overriding factors for making procurement decisions: "Ultimately, the software that best meets the needs and mission of the Department should be used, regardless of whether the software is open source."

However, a variety of misconceptions about OSS are identified by DCIO Wennergren as constraining the adoption of OSS solutions in some DoD departments. These misconceptions include concerns about reviewing code, supporting the software, and making the source code publicly available. But none of these concerns are warranted, the memo asserts.

For example, Public Domain Software Control, DCPD-1, in DoD Instruction 8500.2, "Information Assurance (IA) Implementation," is sometimes cited as restricting the use of OSS. The control states,

Binary or machine executable public domain software products and other software products with limited or no warranty such as those commonly known as freeware or shareware are not used in DoD information systems unless they are necessary for mission accomplishment and there are no alternative IT solutions available.

This control protects against the procurement of software when the Government does not have access to the original source code, making it difficult or impossible to review, repair, or extend the software. But CIO guidance points out that because the government *does* have access to the original source code of open source software, these terms do not apply. For this reason, the DoD Open Source Software FAQ, available through the DoD web site and Intelink, states "…do not use the terms 'freeware' or 'shareware' as a synonym for 'open source software'."

Another concern about OSS is that a lack of appropriate maintenance and support presents an information assurance risk. But this is true for all software, open source or closed source. System and program managers, and ultimately designated approving authorities (DAAs), are responsible for ensuring that a plan for software support is in place and adequate for mission needs before approving the use of *any* software.

The misconception that OSS should not be integrated or modified for use in classified or other sensitive DoD systems is also challenged. The memo notes: "…many open source licenses permit the user to modify OSS *for internal use* [emphasis provided] without being obligated to distribute source code to the public."

Still, federal agencies are required to disseminate new software as widely as possible. Because software source code and associated design documents are defined as "data" by DoD Directive 8320.02, they are to be shared across the DoD to support mission needs. OSS licenses actually make it easier to share these components, providing even better support for the DoD's network-centric data strategy. Therefore, it is up to the project manager, program manager, or other comparable official to understand how the Department intends to use and redistribute any DoD-modified code and the specific requirements of the governing OSS license.

Acting Assistant Secretary of Defense for Networks and Information Integration (ASD(NII)) and DoD CIO Cheryl Roby calls information "our greatest strategic asset." To transform the DoD into a network-centric organization, roadblocks to that information must come down. DCIO Wennergren believes achieving a net-centric force is "much more about culture change than technological change." Part of that culture change means accepting OSS as a viable software solution for meeting mission needs. The adoption of more open-source software projects by federal agencies could mark an important step along the road to an information-centric future for the DoD.

Raising the Bar in Operating System Security:

SELinux and OpenSolaris FMAC

Abstract

Over the past several years, the Security-Enhanced Linux (SELinux) reference implementation of the Flask security architecture has undergone a rapid evolution in its capabilities and maturity thanks to a large and growing developer and user community. SELinux has also influenced a wide range of related work in other operating systems, hypervisors, and applications. In 2008, a new project was started to bring the same Flask security architecture demonstrated in SELinux to the OpenSolaris™ operating system via the OpenSolaris Flexible Mandatory Access Control (FMAC) project. These efforts have fundamentally changed the terms of debate about operating system security and ushered security features previously limited to separate niche products into the mainstream. This article describes the major advances and changes in SELinux that have occurred during the last several years; summarizes other related work that has flowed out of the SELinux project; and introduces the goals, design, and status of the OpenSolaris FMAC project.

Introduction

Security-Enhanced Linux (SELinux) was developed by the National Information Assurance Research Laboratory (NIARL) of the National Security Agency (NSA) starting in 1999 and was first released to the general public via the nsa.gov web site in December 2000. SELinux was created by NSA as a reference implementation of the Flask security architecture for flexible mandatory access control (MAC) in order to show how such controls could be added to a mainstream operating system and to demonstrate the value of MAC [1]. SELinux was intended to serve both as a technology transfer vehicle for encouraging adoption of flexible MAC into mainstream operating systems and as a research platform for advanced security research and development. Prior to the release of SELinux, MAC was only available in separate "trusted" operating system products and was limited to fixed hierarchical security models that were unable to express many kinds of real security goals.

The public release of SELinux drew the interest of both advanced Linux users and the Linux kernel developers, which led to an invitation to present SELinux at the Linux kernel developer

summit in March 2001. The resulting discussion at that summit led to the creation of the Linux Security Modules (LSM) project, an open source project to create a common security framework in the Linux kernel that could support a variety of security models. During the next couple of years, the SELinux developers served as core contributors to the development of the LSM framework and re-architected SELinux to use the LSM framework. The LSM framework began to be merged into the mainline Linux kernel in 2002, and the remaining portions of the framework and the SELinux security module were merged into the mainline Linux 2.6 kernel series by the end of 2003.

Even prior to its integration into the mainline Linux kernel, advanced Linux users had begun packaging SELinux kernel, policy, and application support for multiple Linux distributions so that they could use SELinux for protecting their own systems. SELinux packages for the Debian GNU/Linux distribution were made available as early as 2001, and the Hardened Gentoo project (a security-focused subproject of the Gentoo Linux distribution) began including SELinux support in 2002. The growing developer and user community around SELinux and the efforts to bring SELinux support into the mainline Linux kernel drew the interest of Red Hat, Inc., which began work to fully integrate SELinux support into its Linux distributions in 2003, starting with their new community-based Fedora distribution. The SELinux code was first included in the Fedora Core 2 release in May 2004, eliminating the need for separate patches for the kernel and applications. The introduction of a security policy configuration focused on confining specific network-facing services such as the Apache web server and the BIND domain name server made it possible to enable SELinux by default in the Fedora

Core 3 release in November 2004. This security policy configuration was called the "targeted" security policy because it applied SELinux to protecting specific services (i.e., the "targets") that were likely points of attack into the system.

The Fedora SELinux integration work and the resulting community testing and refinement of SELinux formed the basis for including SELinux in the commercially supported Red Hat® Enterprise Linux® product. Red Hat Enterprise Linux 4, released in February 2005, shipped with SELinux as a default-enabled security feature providing out-of-the-box confinement of over a dozen system services. This release represented the first inclusion and use of MAC in a mainstream commercial operating system. MAC was no longer limited to separate "trusted" operating systems and had become a general-purpose security feature. The inclusion of MAC in a mainstream commercial operating system set the stage for the rapid advances in SELinux that have occurred since 2005.

SELinux: 2005–present

Policy technology advances

Over the past several years, a new generation of policy technology has been developed and deployed for SELinux. The advances in policy technology have included the introduction of the reference policy, the development of loadable policy module support, the creation of policy management infrastructure, and the convergence of strict and targeted policies.

SELinux was originally released by NSA with a small example policy

> "Linux security experts are reporting a growing list of real-world security situations in which the US National Security Agency's SELinux security framework contains the damage resulting from a flaw in other software."
>
> *Don Marti , LinuxWorld.com*

configuration to demonstrate the concepts and the value of flexible MAC. Early adopters of SELinux used that example policy as a base and began contributing changes and additions to it, leading to very rapid growth in its coverage of different applications but at a cost in terms of understandability and ease of customization. NSA sponsored work by Tresys Technology to undertake a re-design of the base policy for SELinux, with a focus on modularity, understandability, tool support, and customization. This work has yielded the SELinux reference policy, which has supplanted the original example policy as the standard base policy for all modern Linux distribution releases that support SELinux, starting with the Fedora Core 5 release in March 2006.

The reference policy was also designed to take advantage of a new feature in the SELinux policy toolchain that was also being developed by Tresys Technology in the same timeframe: support for loadable policy modules. The original SELinux policy configuration and compiler were "monolithic." That is, in order to make any substantive change to policy beyond a few specific forms of customization (e.g., booleans, local file contexts), one needed to obtain a complete policy source tree, make corresponding changes to the source files, and rebuild the entire policy into the binary form required by the kernel. Loadable policy module support was developed to enable individual policy modules to be built and packaged separately from one another. This mechanism has enabled users to easily create local policy modules as needed for site customization, and it has enabled software developers to easily package policy for their applications. Loadable policy module support was also first deployed in Fedora Core 5.

While the loadable policy module support was being merged into the upstream SELinux userland, a new

software library, *libsemanage*, was developed jointly by Red Hat and by Tresys Technology to provide a standard API and infrastructure for managing policy. This library provides a programmatic interface for making changes to policy, as opposed to having to manually edit text files, and provides support for a wide range of local customizations to policy. Front-end tools such as *semodule* and *semanage* were created to enable users and higher level tools to perform policy management tasks. This library and the initial front-end tools also first appeared in Fedora Core 5.

A practical compromise made early in the Fedora SELinux integration was to create a separate "targeted" policy configuration that focused on protecting network-facing services and left ordinary user sessions unrestricted and use that policy as the default so that SELinux could be enabled by default without disrupting users. The complete example policy with significantly more coverage of services and applications and support for user roles became known as the "strict" policy configuration, and this strict policy configuration was not well supported and required significant expertise to successfully install and use. However, this compromise made it possible to incrementally expand the coverage of the targeted policy in each new release through the community testing and feedback process since SELinux was enabled by default. With the introduction of the reference policy, both the strict and targeted policy variants were built from the reference policy sources based on a single tunable setting.

As a result, the targeted policy has grown from covering over a dozen services in the earliest release to covering over two hundred applications in modern releases. The last significant difference between the targeted and strict policies was eliminated starting with the Fedora 8 release in November 2007, when support for confining users was introduced in targeted policy. The Fedora 9 release in May 2008 used this support to define several user roles available by default

and to support a kiosk mode of operation where the user session is highly restricted and is completely purged of state after each session. As a result, the targeted policy and the strict policy have converged and there is no longer a separate strict policy. Administrators can largely obtain the behavior of the strict policy by mapping users to confined roles, and they can optionally remove the unconfined policy module entirely, although this last step can be destructive to running processes and requires some care to do safely.

Improved usability

Based on user feedback, the advances in policy technology described above have greatly improved the user experience of SELinux by enabling users to solve many of the problems that they encounter. In particular, the loadable policy module support and the management tools have enabled users to perform local customizations of policy to fit their particular needs and have enabled developers to ship customizations for their applications.

Figure 1: system-config-selinux screenshot

Figure 2: setroubleshoot screenshot

Several graphical tools have also been developed in recent years to assist users with different aspects of SELinux. These tools include *system-config-selinux*, *setroubleshoot*, and the SELinux Integrated Development Environment (SLIDE). The first two tools were developed by Red Hat and first included in Fedora Core 6 in October 2006. The SLIDE tool can be freely downloaded from the Tresys Technology open source server, *http://oss.tresys.com/*, and was included in the Fedora 9 release in May 2008.

A graphical front-end to the *semanage* functionality, *system-config-selinux* allows the administrator to easily see and modify the current enforcing status, policy booleans, label assignments for files and ports, and user authorizations. It also provides simple support for managing the set of loaded policy modules. Recent versions of the tool (see Figure 1) also include a policy wizard for creating new policy modules.

A service for notifying users of SELinux denials, *setroubleshoot* helps users to diagnose denials and resolve them. It has increased user awareness of SELinux and enabled users to identify and solve common kinds of configuration errors. The tool can be configured to display alert popups to the user on the desktop, or alerts can be handled via system logs or email notifications. See Figure 2 for a screenshot of *setroubleshoot*.

SLIDE is an Eclipse plug-in to provide a graphical user interface for policy developers with the conventional features of an integrated development environment, such as policy creation wizards, interface completion and searching, and policy syntax highlighting. Recent versions of SLIDE (see Figure 3) have incorporated support for remote policy debugging and integration with policy analysis tools.

Enhanced security functionality

The core security functionality of SELinux has undergone significant enhancements and improvements since 2005. These enhancements have included extended security audit functionality,
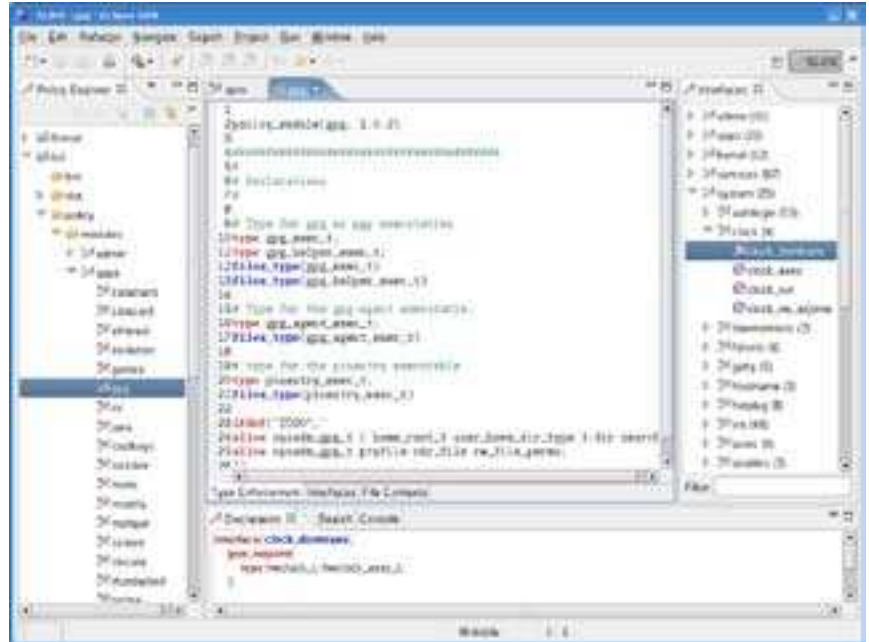
Figure 3: SLIDE screenshot

enhanced Multi-Level Security (MLS) support, new network access controls, and labeled networking support along with numerous other smaller changes.

As part of the work to enable Linux to meet the Labeled Security Protection Profile requirements, SELinux was fully integrated with the Linux audit subsystem, enabling audit to include and filter based on security contexts. Likewise, the optional MLS model of the SELinux security server was enhanced and enabled by default, and user space support for MLS requirements was developed. This work was done with the help of a wide range of contributors from HP, IBM, Red Hat, and Trusted Computer Solutions (TCS).

Red Hat developed a new mechanism for flexible network access controls called SECMARK, which combined the power of the Linux packet filtering framework with SELinux policy. Developers from IBM, TCS, and HP created and integrated two independent implementations of labeled networking mechanisms, labeled IPSEC and NetLabel/CIPSO. These mechanisms enable SELinux protections to be applied across network communications.

Improved performance and scalability

The original SELinux implementation included an Access Vector Cache (AVC) to minimize the need to perform expensive security computations on each operation and sought to keep overheads to reasonably low percentages, but did not specifically engage in any detailed performance optimizations. The original SELinux implementation also only dealt with ensuring safety on systems using an SMP (symmetric multiprocessing) version of the Linux kernel through the use of coarse-grained locking. This approach did not scale well on large SMP systems. Since the original inclusion of SELinux in Linux distributions, a number of performance and scalability improvements have been developed and integrated.

An engineer from NEC Corporation undertook work to enable SELinux to scale well on large SMP systems. He replaced the coarse-grained lock of the AVC with a scheme known as Read-Copy-Update (RCU), enabling SELinux to achieve near-perfect scalability. This made the deployment of SELinux practical on large systems.

As the default targeted policy grew in its coverage of services, the amount of kernel memory used by the policy was increasingly becoming too high and beginning to cause problems for users. As a result, the SELinux core developers discussed approaches to improve memory

use, and a set of memory optimizations were implemented by NSA that radically reduced the kernel memory use by a factor of 20X.

A number of memory and performance optimizations have been developed in recent years by contributors from the Japanese SELinux community working on using SELinux on embedded systems, including contributions from NEC and Hitachi Software among others. Further optimizations to the policy data structures have yielded significant improvements in kernel memory use. The revalidation of read and write permission on individual read and write system calls was optimized to deal with significant overheads on the SuperH architecture, improving overhead by a factor of around 10X.

Meeting security criteria

In 2007, Red Hat Enterprise Linux 5 was validated against the Controlled Access Protection Profile (CAPP), the Labeled Security Protection Profile (LSPP), and the Role-Based Access Control Protection Profile (RBACPP) at Evaluation Assurance Level 4+ on HP and IBM hardware. This was the result of a collaborative effort among HP, IBM, Red Hat, and TCS that leveraged SELinux to provide the labeled security and role-based support. This work led to several improvements to SELinux, including improved audit, MLS, and labeled networking. SGI has also achieved validation of Red Hat Enterprise Linux 5 on its hardware in 2008. These validations represent the first time that a mainstream commercial operating system product has been validated against such criteria, which in the past have been limited to separate "trusted" operating system products. These evaluations were also distinctive in that the companies that sponsored the evaluations worked together to develop the necessary functionality and shared the resulting documentation and test suites despite being competitors.

The Systems and Network Analysis Center (SNAC) of NSA developed and released their first-ever security

configuration guide for Linux in 2007, the *Guide to the Secure Configuration of Red Hat Enterprise Linux 5*. Along with many other topics, the guide describes the benefits of SELinux for security and explains how to perform basic configuration and troubleshooting of SELinux. This guide joins the other configuration guides produced by the SNAC over the years for a wide variety of operating systems and is available from *http://www.nsa.gov/ia/guidance*.

The Certifiable Linux Integration Platform (CLIP) is a specific configuration of Linux and associated evidence designed to meet several established security requirements, including the Protection Level (PL) 4 requirements from the Director of Central Intelligence Directive (DCID) 6/3, "Protecting Sensitive Compartmented Information within Information Systems" and the High Impact requirements from the National Institute of Standards and Technology (NIST) Special Publication 800-53, "Recommended Security Controls for Federal Information Systems." CLIP defines a specific configuration of SELinux to provide the foundation for hosting security-relevant applications by ensuring that the underlying assumptions made by those applications are enforced by the operating system. In particular, CLIP leverages SELinux in order to enforce the strong separation of processes and data, support different user roles, and ensure that application security mechanisms are tamperproof and cannot be bypassed. The CLIP project is sponsored by NSA's Custom Solutions Group and is being developed by Tresys Technology. CLIP can be downloaded from *http://oss.tresys. com/projects/clip*.

Growing adoption, use, and community

For the past four years, SELinux has been a default-enabled security feature in the Fedora and the Red Hat Enterprise Linux distributions, providing out-of-the-box confinement of an increasing number of system services. The improved usability of SELinux has enabled users

to expand their use of it and to directly solve problems. The anecdotal evidence of improved user experience from public mailing list discussions is further reinforced by statistics being collected by the Fedora project, which began to collect information about SELinux status starting with the Fedora 8 release. The majority of Fedora systems reporting into the Fedora project show that users keep SELinux enabled.

In addition to Fedora and Red Hat distributions, SELinux has continued to make advances in adoption in other Linux distributions. The Hardened Gentoo project has continued to support SELinux in the Gentoo Linux distribution and to integrate newer SELinux features. The Debian GNU/Linux distribution began including SELinux support in the Debian 4.0 release. The Ubuntu distribution began including minimal SELinux support in the Ubuntu 8.04 release, which was then further enhanced in the Ubuntu 9.04 release. Novell began including basic SELinux support as an optional feature in SUSE Linux 11.1.

The benefits of SELinux for mitigating vulnerabilities in software are increasingly being recognized. An article by Don Marti published on LinuxWorld. com in February 2008 stated, "Linux security experts are reporting a growing list of real-world security situations in which the US National Security Agency's SELinux security framework contains the damage resulting from a flaw in other software [2]." In discussing the migration of their mission-critical trading platform to Linux, Steve Rubinow, the Chief Information Officer (CIO) of the New York Stock Exchange (NYSE) Euronext was quoted as saying, "We are very security conscious because we have to be….We maintain the security of our systems by relying on the SELinux features within Red Hat Enterprise Linux [3]."

SELinux has also served as a secure foundation for a number of secure solutions developed for the government. These systems include the NetTop® system originally prototyped by NIARL

and later productized by HP along with several derivative systems. It also includes the TCS Secure Office® Trusted Thin Client system. A number of Cross Domain Solution (CDS) systems have been developed by NSA and by other organizations that leverage SELinux to enforce separation and to ensure the assured invocation of the CDS application. As mentioned earlier, SELinux is also being leveraged by the CLIP project.

Along with growth in its user community, SELinux has experienced significant growth in its developer community since 2005. Developers from HP, Hitachi Software, IBM, NEC, NSA, Red Hat, Tresys Technology, and TCS along with many individual developers have worked collaboratively to enable SELinux to advance rapidly in its feature set and maturity.

Platform for advanced R&D

In addition to serving as a technology transfer vehicle for encouraging adoption of flexible MAC by industry, SELinux has also served as a useful platform for advanced research and development. By providing a base system that supports flexible MAC and exports interfaces to support security-aware applications, SELinux enables research to proceed in understanding MAC in a complete system, from the low-level operating system up through infrastructure layers to the end-user applications.

Securing the desktop environment experienced by typical users is one area of active research. This area is particularly challenging to secure due to the tight coupling of applications typical in such environments and the lack of consideration to any security boundary between processes within a desktop session. Addressing these challenges is critical in order to be able to protect against exploitation of flaws in commonly used desktop applications such as browsers and mail clients, so that a flaw in a single program does not expose all of the user's data to risk. To date, work has been done by NSA to implement the D-BUS message service, the GConf configuration system, and the X Window System server with the necessary support for applying flexible MAC to their objects and operations [4, 5]. Work is ongoing by NSA to develop library support for these extensions, address other components of the desktop infrastructure, and assist in developing policy for the X server that supports simple security goals. Future work includes addressing performance challenges, refining the controls based on experience with real applications, securing the direct rendering interface, providing trusted input and display, and integrating with desktop applications.

Beyond the desktop, a wide range of application security research is leveraging SELinux as a base platform and as an architecture for providing flexible MAC services to higher layers. This includes the SE-PostgreSQL project, an effort by NEC Corporation to develop flexible MAC for database objects and transactions in the Postgres database management system. Research has also been performed by NSA into enforcing Risk Adaptive Access Control (RAdAC) by leveraging the SELinux operating system functionality to protect and isolate an application policy enforcer and by using the Flask architecture and user space security server to provide policy decisions and revocation support [6].

Enabling secure file sharing among networked or distributed systems is another area of active research being led by NSA with support from SPARTA, Inc. This effort requires addressing challenges posed by systems with potentially different security policies that need to share data securely as well as providing the basic mechanisms for conveying security attributes for processes and files across the network. Given the common use of such networked file systems in enterprise environments, enabling flexible MAC to be effectively applied to such file systems is likewise a crucial challenge. Experimental extensions to the NFSv4 protocol have been proposed and prototyped, and work is ongoing to standardize the protocol changes and to get the implementation into a form acceptable to the Linux developer community. Future work will include dealing with heterogeneous policies.

While flexible MAC provides new capabilities for improved security, it also introduces its own set of challenges, including policy scalability and usability. Hence, research by NSA with support from Tresys Technology is ongoing into how to create an abstract layer for policy and how to more closely link the enforcement, debugging, and development of policy to enable users to more effectively develop, debug, and understand policy. Research is also underway at NSA into more advanced policy language features to facilitate policy customization and extension.

Policy management continues to be an area of active investigation. Early work in this area by Tresys Technology and Red Hat has yielded the current policy management infrastructure and tools such as *semanage*. Research by Tresys Technology, which was sponsored by NSA, has also yielded experimental prototypes of a policy management server to support fine-grained access control over the policy itself and of policy management infrastructure to support management of collections of systems. Work has recently started at Penn State University to investigate how to manage policy for virtualized environments with different collections of policy enforcing components.

Influencing other systems

The Flask security architecture demonstrated in SELinux has strongly influenced the security of a number of other systems and software components. In the application arena, this has included the D-Bus message bus software, the X Window System, and PostgreSQL, as previously noted, each of which now has a set of flexible MAC controls implemented that can extend the reach of the policy enforcement to their higher level objects and operations. In the virtualization arena, the Flask architecture has been applied to the Xen hypervisor, yielding the Xen Security Modules (XSM) framework

and the Xen Flask security module developed by NSA, enabling enforcement of policy over virtual machines and their interactions.

The Security-Enhanced BSD (SEBSD) and Security-Enhanced Darwin (SEDarwin) projects demonstrated that the Flask architecture could also be applied to other operating systems. Although SEBSD and SEDarwin are not integrated into their respective mainstream operating system distributions, they helped to drive the requirements for the MAC framework that can be found today in mainstream FreeBSD® and in MacOS X operating systems. They also proved that the architecture was applicable to a variety of operating systems and provided an alternative reference implementation from which others can learn.

OpenSolaris FMAC: origin and goals

In late 2007, NSA and Sun Microsystems, Inc., began a dialogue about integrating support for the Flask architecture into the Solaris™ operating system. This dialogue led to the launching of the Flexible Mandatory Access Control (FMAC) project on OpenSolaris.org in March 2008. The project is a joint effort among NSA, Sun, and the OpenSolaris developer community to bring support for flexible MAC to the OpenSolaris operating system environment.

Unlike Linux, where there was no support for MAC at all prior to the integration of SELinux, the Solaris operating system has an existing MAC solution. Prior to Solaris 10, this functionality was provided by Sun via a separate product, the Trusted Solaris operating system. Like other trusted operating systems of its genre, Trusted Solaris was limited to a fixed MLS security model and tended to lag behind the latest release of Solaris due to the additional engineering and evaluation requirements. In Solaris 10, some of the security functionality of Trusted Solaris, such as support for roles and privileges, was

integrated into the main Solaris product and released as part of OpenSolaris. The MLS support was redesigned around the Solaris "zones" mechanism and provided as an optional set of extensions to Solaris known as Trusted Extensions (TX), which have also been subsequently integrated into OpenSolaris.

Solaris zones are a mechanism for lightweight virtualization; they provide an illusion of multiple virtual operating system instances while sharing a single kernel by placing groups of processes and objects into separate "zones" and isolating them from one another. Since TX relies on the zones mechanism, it is limited to per-zone security labels (i.e., all processes and objects within a zone share the same security label). This represents a change from the prior Trusted Solaris product and a difference from SELinux, both of which support per-process and per-object security labeling. TX is also presently limited to a fixed MLS model like its predecessors.

FMAC aims to address these limitations by supporting per-process and per-object labeling and permission checks, and by introducing flexible MAC support to OpenSolaris that can support a wide range of security models. The zone-based mechanism will still be useful as a way of providing coarse-grained isolation and namespace separation, while FMAC will be used in a complementary fashion to provide intra-zone protection, hardening of the global zone, and control over cross-zone channels. In this manner, FMAC and TX should be able to complement one another and ultimately form an integrated solution.

FMAC also aims to complement the existing Solaris privilege and Role-Based Access Control (RBAC) mechanisms. Just as SELinux provides a way to control the use of Linux superuser capabilities based on policy, FMAC will provide a way to control the use of Solaris privileges based on policy. This control will include the ability to bind privileges to specific processes and programs, to limit the use of privileges by a given process to

specific objects, and to protect privileged processes from untrustworthy inputs, just as in SELinux. Unlike Linux prior to SELinux, Solaris has an existing RBAC mechanism, but at the present time the Solaris RBAC mechanism is primarily enforced by trusted applications, with the kernel only aware of the privilege model. FMAC offers a means of binding the Solaris role construct to processes and directly enforcing RBAC restrictions in the kernel, strengthening the existing mechanism.

While complementing these existing Solaris security mechanisms, FMAC will preserve existing Solaris interfaces and provide full compatibility for applications, just as SELinux provided full compatibility for Linux applications. FMAC will also provide a set of new APIs that will support security-aware applications, and these APIs will provide the same semantics as the corresponding SELinux APIs so that security-aware applications can be written portably to run on either SELinux or OpenSolaris FMAC.

Ultimately, the OpenSolaris FMAC project will provide a wider set of platforms that support the Flask architecture for flexible MAC and will expand the developer and user community for Flask. It should also help encourage independent software vendors (ISVs) to improve application-level support for flexible MAC and to provide policies for their applications.

FMAC status

The initial FMAC code base was contributed by NSA to OpenSolaris based on a version of the Flask code that predated any involvement by the Linux community. This code was then integrated into the OpenSolaris code and adapted by John Weeks, a Sun engineer who is the co-lead of the FMAC project. This code was first released publicly as an Alpha 1 release on the FMAC project web site in May 2008. When built, it produced a policy compiler and a kernel capable of loading the resulting policy into the security server.

Since that first release of FMAC, joint development by NSA and Sun Microsystems has proceeded rapidly. Support for new system calls and utilities and for per-process security labeling was introduced during the summer of 2008, leading to an Alpha 2 release in early September. Shortly thereafter, prototype support for per-file security labeling in the ZFS file system was introduced, which paved the way for supporting security context transitions upon program execution and for performing a basic set of process and file mandatory access control checks. This work produced a basic working example of how flexible mandatory access controls could be applied to an OpenSolaris system. This functionality along with subsequent enhancements to support labeling in the TMPFS file system and improve the Access Vector Cache (AVC) interfaces and implementation in FMAC was included in the Alpha 3 release made in late October 2008.

The next two major areas of focus for FMAC integration are privileges and RBAC. Significant work also remains to label and control other objects and operations provided by the Solaris kernel, create a complete example policy configuration, and integrate support for FMAC fully into user space. More advanced development and collaboration with the SELinux project in areas such as securing the desktop, policy usability and management, and labeled NFS will likely follow as FMAC matures.

Conclusion

The SELinux project has brought flexible MAC into the mainstream, achieving success both as a technology transfer vehicle and as a platform for advanced research and development. It has influenced a wide range of systems and software components, as shown most recently in the OpenSolaris FMAC project. The developer and user community that has arisen around the core ideas embodied in SELinux and OpenSolaris FMAC gives confidence that this technology will continue to be preserved and built upon in future computing systems, providing a solid foundation for addressing the threats posed by flawed and malicious applications. The advances in usability, performance, and functionality over the past several years have made these benefits far more accessible to end users.

Resources

NSA SELinux web site, http://www.nsa.gov/research/selinux

SELinux project wiki, http://selinuxproject.org

Tresys Open Source Server, http://oss.tresys com

OpenSolaris FMAC web site, http://opensolaris.org/os/project/fmac

References

[1] Loscocco P, Smalley S. Integrating Flexible Support for Security Policies into the Linux Operating System. In: Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference; June 2001.

[2] Marti D. A seatbelt for server software: SELinux blocks real-world exploits. Available from: http://www.linuxworld.comnews/2008/022408-selinux.html

[3] Red Hat. NYSE Euronext Chooses Red Hat Solutions for Flexibility and Reliable, Fast-Paced Performance. Available from: http://customers.press.redhat.com/2008/05/12/nyse/

[4] Walsh E. Application of the Flask Architecture to the X Window System Server. In: Proceedings of the 2007 SELinux Symposium; March 2007.

[5] Carter J. Using GConf as an Example of How to Create an Userspace Object Manager. In: Proceedings of the 2007 SELinux Symposium; March 2007.

[6] Gregory M. Using the Flask Security Architecture to Facilitate Risk Adaptable Access Controls. In: Proceedings of the 2007 SELinux Symposium; March 2007.

Providing a Secure Foundation for Applications with the Certifiable Linux Integration Platform

The needs of the national security community frequently require custom computing solutions; however, current development practices result in each solution requiring an individualized secure foundation. Without a common foundation, each computing solution must then be developed and certified separately. The Certifiable Linux Integration Platform (CLIP) provides this common foundation for secure solutions and is targeted to decrease the time and associated cost spent on development and certification. In this article, we describe the CLIP project and highlight what CLIP provides to support custom solution development, particularly solutions that must be certified.
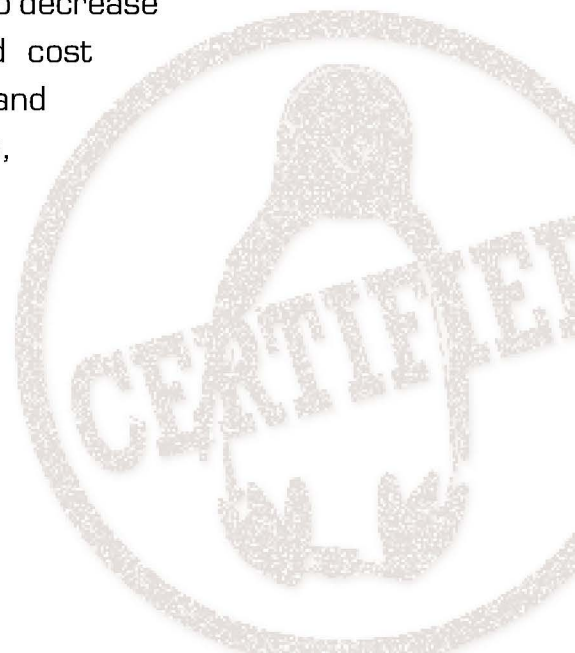
What is CLIP?

CLIP is an effort pioneered by the National Security Agency's Assured Information Sharing Technologies and Products Office with the goal of decreasing the time and cost associated with certifying and deploying trusted solutions. This project helps achieve NetCentric Security Technologies' mission to provide technical information assurance (IA) guidance and to support the system security engineering process. The CLIP project provides an open source base that multiple projects can easily utilize as a resource. More specifically, CLIP provides

- Underlying system configuration,

- Initial application configuration,

- Updated Security-Enhanced Linux (SELinux) policy with the goal of creating a more secure environment,

- Updates and additional packages necessary to meet certain requirement sets, and

- Artifacts describing how the system maps to requirement sets that can be used as part of the evidence for certification.

Need for a secure foundation

When constructing solutions vital to national security, system developers should ensure compatibility by properly configuring their custom applications with all other applications running on the operating system and with the operating system itself. However, developers frequently have neither the time nor the operating system expertise to properly configure all parts of the system. And, even when properly configured, each system is often created and used for a single project and rarely shared among peers. This lack of sharing creates an environment where time, effort, and money must be spent during every development to repeat the procedure of locking down the system and its applications.

Certification and Accreditation

Certification and Accreditation (C&A) is an important and necessary part of the deployment of any security system. Every time a system is accredited, it must go through the certification process to verify that it meets all of its security and functional requirements. This process requires that developers spend time creating system documentation and mapping the documentation to requirement sets. The certification teams must also create tests that are then used to verify the developers' claims. Testing and verification is currently performed for each and every system that goes through the C&A process because no common configuration is used for all systems.

Reliance on proprietary hardware and software

When creating any custom solution, the developer faces the hurdle of obtaining access to the operating system source code. In the past, many solutions were created on closed source operating systems that limited the developer's access to the source files. The developer was subsequently limited to only the list of documented calls, which may or may not have been the full set of actual calls. In this model, if developers found a problem, they had to rely on the vendor to provide a solution.

The use of proprietary hardware is another limiting factor in today's solutions. Custom solutions use operating systems that must run on proprietary hardware and cannot easily be ported. As the hardware ages, parts that are no longer manufactured must be replaced. This environment forces the developer to purchase duplicate sets of hardware to ensure long-term support.

CLIP explained

CLIP toolkit

The CLIP toolkit can be used by system developers to create a secure starting point when building solutions. Each toolkit is specific to a particular release of a commercially supported operating system. Toolkits have been created for Red Hat Enterprise Linux (RHEL) versions 4 and 5.0 through 5.4. The toolkit for RHEL 6.0 is currently under development.

The toolkits vary in the specific packages they provide, but at minimum each one provides a kickstart file that controls the initial system

configuration, a CLIP package manager (RPM) that installs CLIP specific utilities, and an updated SELinux policy RPM that hardens the standard reference policy package. The RHEL 4 toolkit provides system updates to enable true role separation. The RHEL 5 release adds to the previous release with support for labeling packets.

Each piece of the toolkit can be used separately if a developer chooses to do so, allowing CLIP to meet the needs of the dynamic environment in which systems are deployed.

SELinux as a basis

At a minimum, a secure foundation requires security mechanisms enforced by the operating system. SELinux provides the basis for this secure foundation.

SELinux [1] is an implementation of Flask [2,3], a flexible and fine-grained mandatory access control (MAC) architecture, in the Linux kernel. The architecture separates the policy decision point, provided by the security server, from the policy enforcement point, implemented by the LSM (Linux Security Modules) framework [4,5].

The Flask architecture [6,7] provides flexibility in its support for security policies, thereby providing mechanisms to support a wide variety of real-world security policies. A security context is attached to every object on a system (e.g., files, processes, network packets). The security policy defines allowed access by *subject security contexts* to *object security contexts* with sets of permissions stored in access vectors. To minimize the performance impact of consulting the security policy, a security decision caching mechanism called the access vector cache (AVC) is used.

SELinux implements a combination of mandatory access control mechanisms to provide maximum flexibility and usability: type enforcement (TE), role-based access control (RBAC), and multi-level security (MLS). Type enforcement [8] has been explored for many years (e.g., Domain and Type Enforcement [9,10,11] and Distributed Trusted Operating System (DTOS) [12]) as an effective and flexible MAC mechanism. In type enforcement, all entities on a system are given a type, and every object or subject with the same type is treated identically. Access decisions are made based on the permissions granted to each type. Role-based access control [13] provides a complementary mechanism to type enforcement in which access is granted based on roles assigned to users. Multi-level security provides a means to process data with different sensitivities or levels using the Bell-LaPadula (BLP) model [14].

SELinux is available in a number of Linux distributions, including Fedora, Gentoo, and Debian, as well as the commercially available Labeled Security Protection Profile (LSPP) Evaluation Assurance Level (EAL) 4+ Red Hat Enterprise Linux (RHEL). It also has been ported to other operating systems including FreeBSD (Security Enhanced BSD) [15] and Apple's Darwin operating system (Security Enhanced Darwin) [16]. The SELinux Reference Policy [17] provides the basis for security policy on most of these systems.

System configuration

The CLIP installation configures the system to meet the Director of Central Intelligence Directive (DCID) 6/3 Protection Level 4 requirements and the Defense Information Systems Agency (DISA) Unix Security Technical Implementation Guide (STIG) v5r1. These configurations fall into three main categories: application and service installation, access control, and auditing.

To increase security and the ease of administration, CLIP installs only the base applications needed to run and administer the system. By default, it excludes many base applications not required for a functioning system, including applications such as web tools, office tools, and desktop environments. Some services are included in the base installation because of their common usage, but are not needed for the system to function. These services are disabled, leaving only a handful running by default.

CLIP is also configured to greatly restrict user access to the system. Unneeded default accounts are either removed or disabled. Direct administrative (root) access to the system is disabled, requiring logging in as an unprivileged user before privilege escalation. Additionally, all network access is

denied and network parameters are modified to prevent remote attacks. Any services requiring a network connection must be explicitly allowed access on a case-by-case basis.

In addition to user access to the system, CLIP uses both discretionary access control (DAC) and mandatory access control (MAC) mechanisms to restrict all access to files and directories to only the minimal set required to meet the base requirements. CLIP modifies the DAC permissions of many important system files, such as log files, configuration files, and run control scripts, to ensure only privileged users may access them. Using the SELinux policy for MAC enforcement, CLIP further limits and confines a user's ability to view and edit security-relevant files.

To track all changes to the system, CLIP enables auditing and adds many audit rules to record a complete history of all security-relevant system actions. This history includes user logins and logouts, changes to DAC permissions and SELinux labels, unauthorized file access attempts, use of privileged commands, and modification of important system files. To ensure a full audit history is always maintained, any critical error of the audit subsystem will cause an immediate shutdown to prevent any possible breach of information.

New packages

To provide the foundation that meets multiple requirement sets, the CLIP project includes updated and new packages as part of its toolkit. These packages augment the base system and provide the developer additional security features.

Authentication

The National Security Systems Instruction (NSSI) 1253 requirement AC-7 calls for enforced limits for users accessing national security systems and information. These limits include a maximum number of consecutive failed logon attempts to access a network during a set period of time, and how long the user has to wait before trying to log on after being locked out.

The current default module used by the Linux authentication system does not support all the required functionality. Furthermore, the module could not be extended to support this requirement. CLIP created the pam_tally3 module to replace the current module, making it possible for developers to satisfy the requirement.

Archiving

The DCID 6/3 requires that a system create backups of the security labels separately from the objects on the system [19]. A mechanism to accomplish this type of backup did not exist for SELinux based systems. The CLIP project created the extended attribute recovery (xar) utility to satisfy this requirement. The xar utility provides an easy way to backup and restore the security labels of objects on an SELinux system.

Networking

RHEL 5 has the ability to label network packets using labeled IPSec or CIPSO. When operating in an environment that does not support labeled networking, it is useful for a system to be able to dynamically label packets based upon a set of rules such as the network the packet was received from or the protocol of the packet. Linux uses the security markings (SECMark) feature to label packets using IPTables firewall rules. This support was not included in RHEL 5 but was back ported by the CLIP project to enable developers to change the label applied to a packet at runtime on a system.

Future directions

As the landscape of C&A changes, the CLIP project must change with it. The current trend is to provide a mechanism to verify that your current system matches the certified configuration. This verification is done using the Secure Content Automation Protocol (SCAP) and has been deployed on all federal non-national security systems. Future efforts will extend the CLIP project to provide updates to the SELinux policy and the SCAP content necessary to verify that the system configuration matches the requirements.

One long-term goal of the CLIP project is to create a library that will decrease the development and accreditation time by generating the necessary artifacts from a single source. Such a system would allow developers to list the set of requirements that

they must meet for accreditation, and then have the library generate their system's configuration scripts, generate the documentation that shows the scripts meet the stated requirements, and finally generate the SCAP content that could be used for verification of that configuration. The certifiers would have a repeatable set of artifacts allowing them to efficiently determine if a system had met requirements.

References

[1] Mayer F, MacMillan K, Caplan D. SELinux by example. New Jersey: Prentice Hall; 2006.

[2] Loscocco PA, Smalley SD. Meeting critical security objectives with Security-Enhanced Linux. In: Proceedings of the 2001 Ottawa Linux Symposium; 2001.

[3] Loscocco PA, Smalley SD, Muckelbauer PA, Taylor RC, Turner SJ, Farrell JF. The inevitability of failure: the flawed assumption of security in modern computing environments. In: Proceedings of the 21st National Information Systems Security Conference; October 1998.

[4] Smalley S, Vance C, Salamon W. Implementing SELinux as a Linux security module. Rockville (MD): NAI Labs Technical Report; February 2006.

[5] Wright C, Cowan C, Morris J, Smalley S, Kroah-Hartman G. Linux security modules: general security support for the Linux kernel. In: Proceedings of the 11th USENIX Security Symposium; 2002; San Francisco (CA).

[6] Assurance in the Fluke microkernel. Secure Computing Corporation Technical Report; 1999.

[7] Spencer R, Smalley S, Loscocco P, Hibler M, Andersen D, Lepreau J. The Flask security architecture: system support for diverse security policies. In: Proceedings of the 8th USENIX security Symposium; August 1999; Washington (DC).

[8] Boebert WE, Kain RY. A practical alternative to hierarchical integrity policies. In: Proceedings of the 8th National Compuer Security Conference; August 1999; Gaithersburg (MD).

[9] Badger L, Sterne DF, Sherman DL, Walker KM. A domain and type enforcement UNIX prototype. USENIX Computing Systems. Winter 1996;(1).

[10] Badger L, Sterne DF, Sherman DL, Walker KM, Haghighat SA. Practical domain and type

enforcement for Unix. In: Proceedings of the 1995 IEEE Symposium on Security and Privacy; May 1995; Oakland (CA).

[11] Oostendorp K, Badger L, Vance C, Morrison W, Sherman D, Sterne D. Domain and type enforcement firewalls. In: Proceedings of the 13th Annual Computer Security Applications Conference; December 1997; San Diego (CA).

[12] Carney M, Loe B. A comparison of methods for implementing adaptive security policies. In: Proceedings of the 7th USENIX Security Symposium; August 1998; San Antonio (TX).

[13] Sandhu RS, Coyne EJ. Role-based access control models. IEEE Computer. February 1996:38-47.

[14] Bell DE, La Padula LJ. Secure computer systems: mathematical foundations and model. Technical Report No. M74-244.

[15] Vance C, Watson R. Security-enhanced BSD. Rockville (MD): Network Associates Laboratories; July 2003.

[16] Vance C, Miller T, Dekelbaum R. Security enhanced Darwin: porting SELinux to Mac OSX. In: Proceedings of the 2007 Security Enhanced Linux Symposium; March 2007; Baltimore (MD).

[17] PeBenito C, Mayer F, MacMillan K. Reference policy for security-enhanced Linux. In: Proceedings of the 2006 Security Enhanced Linux Symposium; March 2006; Baltimore (MD).

[18] Security Controls Catalog for National Security Systems: Extending National Institute of Standards and Technology Publication 800-53 to national security systems and information. December 2007.

[19] Director of Central Intelligence Directive 6/3: Protecting sensitive compartmented information within information systems. April 2002.

Cryptographic Binding of Metadata
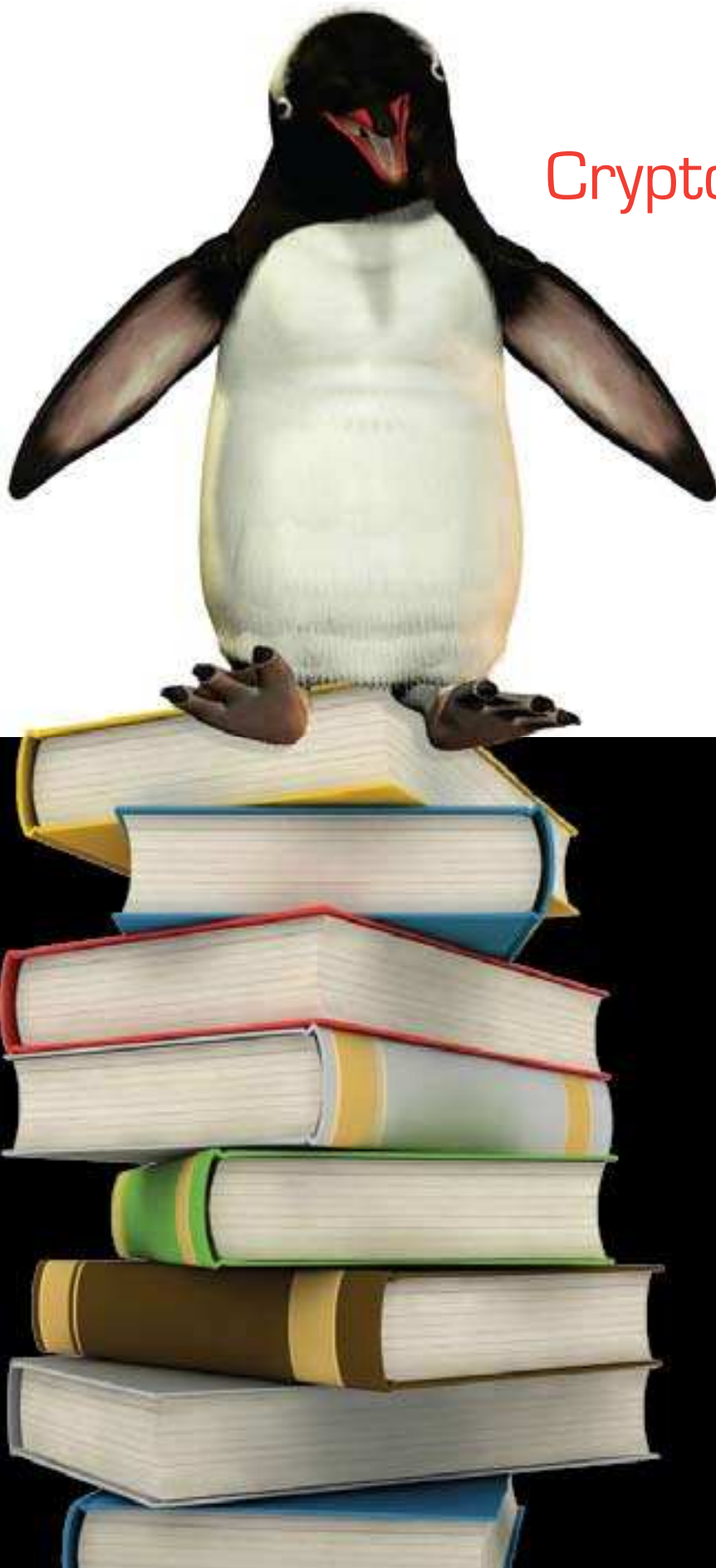
As most people know, metadata is "data about data." It may include security labels and discovery information, as well as user and environmental attributes. Metadata is intended to be used by human consumers or by autonomous processes such as access control mechanisms in the Global Information Grid (GIG), network-centric content discovery services, or automated information dissemination systems. As decisions are made based on metadata content, the assurance provided for the actual metadata must be considered.

In many scenarios, the assurance provided to metadata and to the relationship between metadata and data is essential. Such scenarios range from simple discovery queries to enabling Assured Information Sharing (AIS) through Cross Domain Solutions (CDS).

What is cryptographic binding?

Cryptographic binding provides assurance to the relationship between data and its associated metadata. A binding also ensures that neither the data nor its associated metadata have been maliciously or accidentally modified without detection. The binding does *not* ensure that the original data or metadata is accurate or correct prior to the binding. As the name implies, cryptographic binding uses cryptography as a technique to assert a verifiable relationship over data and its associated metadata. The relationship established with a cryptographic binding is claimed valid if the bound data has integrity and the identity of the binder is authenticated.

How does cryptographic binding work?

Data formats, metadata standards, and cryptography are continually evolving within the Department of Defense (DoD) GIG and the Intelligence Community (IC). For example, metadata to fulfill the needs of the IC is still being defined in many areas. With cryptographic binding depending on these evolving data standards and formats, it is important to establish a flexible and modular binding as well as a validation model that meets the community's needs and can cope with this ever-changing operating environment. The design of cryptographic binding centers on several key assumptions:

- Data and metadata may exist in any discrete format (e.g., XML, HTML, .doc, .xls, .txt, .ppt, .pdf)

- Metadata may exist embedded within data or as a separate file

- Cryptographic binding functions must not modify the data or metadata

- Multiple metadata files may exist for data (e.g., discovery metadata, IA metadata, user and environmental attributes)

- Cryptographic binding functions may exist as embedded applications or distributed services

The cryptographic binding model offers two complementary functions, each with a distinct set of inputs and outputs. First, a binding function, often referred to as the binder, has the sole responsibility of creating cryptographic bindings. The binder accepts the data and metadata files and uses a cryptographic technique to create the binding. The binder produces the asserted relationship as a binding information file (.bif). The validation function, often referred to as simply the validator, accepts the data, metadata, and previously generated .bif files, and applies the same cryptographic technique to verify the integrity and authenticity of the relationship. The validator produces a "valid" or "not valid" response indicating the validity of the binding. Figure 1 illustrates this model for creating and validating cryptographic bindings.

The .bif satisfies the need to create a binding without modifying the data or metadata files. The .bif contains the minimum data required for a validator to verify the integrity and authenticity of the binding. The fields in the .bif file include, but are not limited to:

- Cryptographic value (e.g., digital signature)

- Cryptographic algorithm identifier

- Data hash value, algorithm, and unique identifier

- Metadata hash value, algorithm, and unique identifier

- Binder identity

- Security markings

- Binding method identifier

Cryptographic binding builds upon underlying cryptographic techniques, such as digital signatures, to provide additional services and information. First,
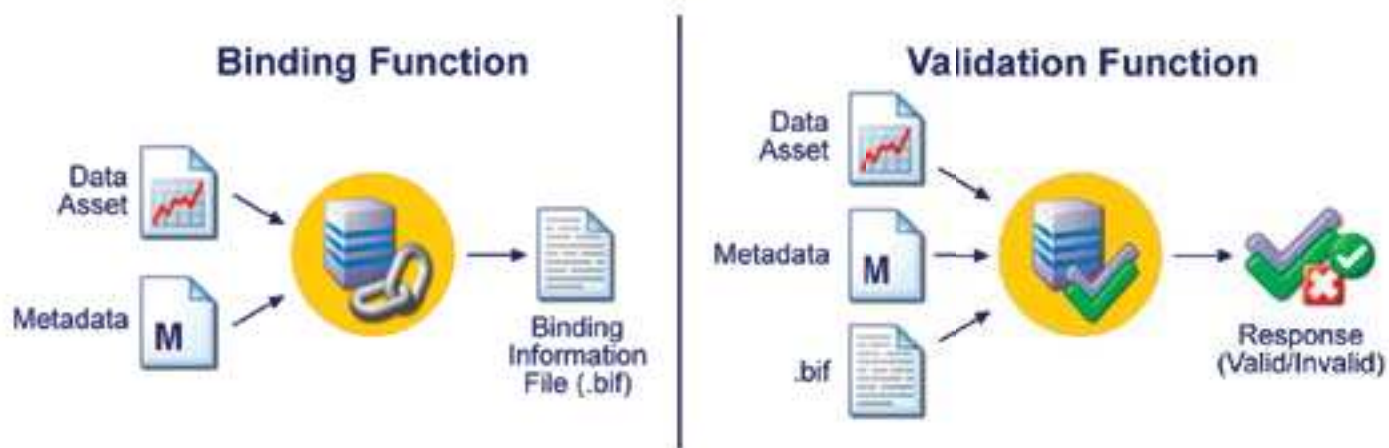

Figure 1: Cryptographic binding and validation service models

although the identity of the binder can be authenticated, the identity of the entity originally claiming that the bound files are indeed related must be captured for traceability and auditing. Second, a cryptographic binding can be thought of as the focal point of data aggregation, possibly bringing an increase of the security level to the binding. For example, imagine a scenario in which a data file contains a list of names and a metadata file contains a corresponding list of departments. Separately, these items are unclassified. However, once the items are cryptographically bound, creating a verifiable relationship, the security level of the information could be increased due to the data aggregation.

This modular architecture separates the functionality from the underlying cryptographic mechanism that provides the integrity and authenticity. Multiple interchangeable binding methods are defined that enable the use of asymmetric cryptography (e.g., digital signatures), symmetric cryptography, and authenticated shared secrets (e.g., secure hashes). Providing these general binding methods enables cryptographic binding to seamlessly incorporate new cryptographic algorithms and techniques.

Proving cryptographic binding concepts

Two cryptographic binding prototypes developed by the NetCentric Security Technologies Division implement the cryptographic binding model and system architecture. These prototypes made use of existing technologies and services to demonstrate the cryptographic binding capability as a system integrated application and an enterprise service. The following are details of each prototype:

Cryptographic binding using XML digital signatures

- Applies to local and distributed architectures
- Implements XML and XML digital signatures (DSIG)
- Uses web services and message transmission optimization mechanism (MTOM)
- Supports RSA 1024-bit encryption and Secure Hash Algorithm 1 (SHA-1)
- Developed using Java 5.0
- Produces a .bif six kilobytes in size

Cryptographic binding using Abstract Syntax Notation 1 (ASN.1) and Cryptographic Message Syntax (CMS)— preferred method

- Applies to local and distributed architectures
- Implements ASN.1 and CMS; studies show ASN.1 is faster to decode than XML
- Implements elliptic curve cryptography (ECC) offering more bits of security using smaller key size and faster algorithmic processing
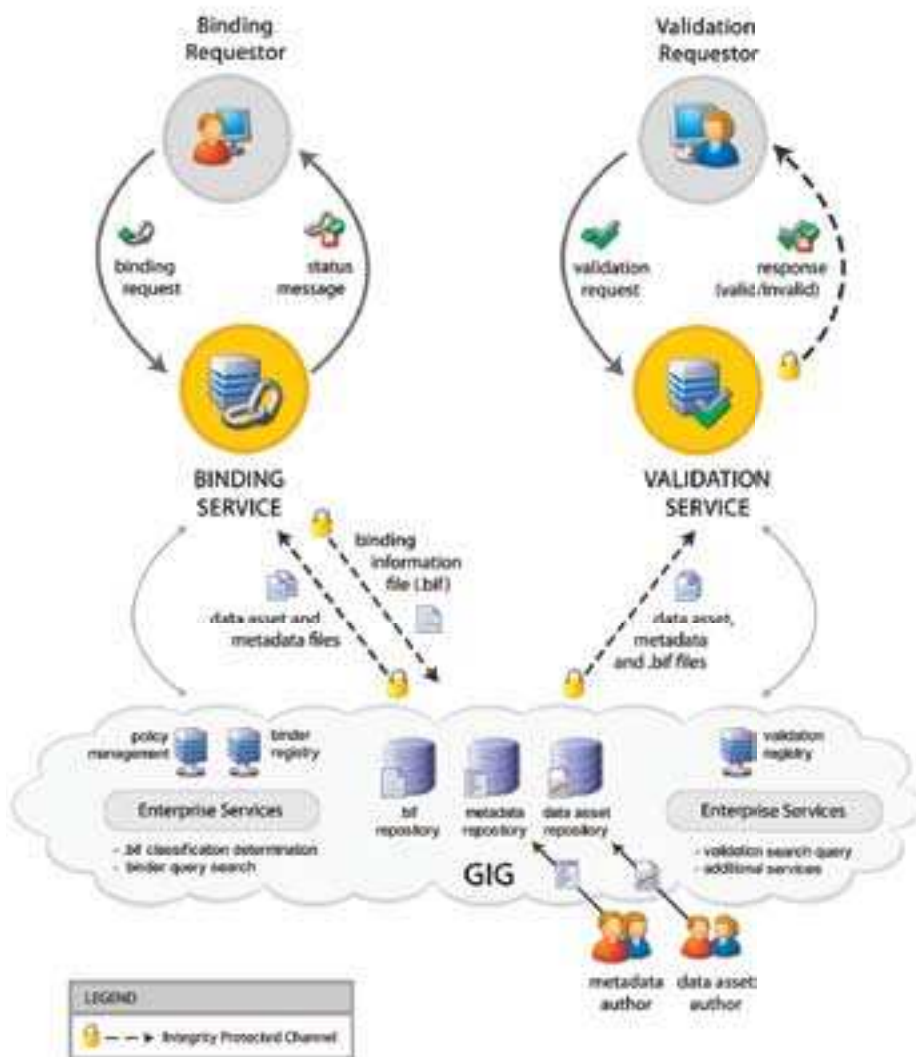- Supports RSA 1024- and 2048-bit encryption

Figure 2: Conceptual view of cryptographic binding

- Supports elliptic curve digital signature algorithm (ECDSA) using 256- and 384-bit prime moduli supporting Suite B Cryptography

- Developed using C++ offering more control over memory allotment than Java, allowing for the binding and validation of larger files

- Produces a .bif 512 bytes in size (a 90% reduction from original XML DSIG .bif)

- Offers additional functionality in new ASN.1 .bif

The XML DSIG cryptographic binding prototype was successfully integrated into several pilot, test, and experiment environments. Community feedback drove the development of the ASN.1 and CMS prototype to improve performance, size, and strength while maintaining core functionality. Therefore, CMS would perform better where the bandwidth is limited and the end unit has minimum processing resource.

Cryptographic binding in future net-centric environments

In the DoD's prospective net-centric GIG, policies will be established throughout the enterprise granting authentication and access to resources. As shown in Figure 2, cryptographic binding will be initiated by a binding requestor—man or machine. In some instances the binding requestor may be the author of the data or metadata. Requestors will be authorized by access control or policy enforcement services. A request is sent by the requestor to the binding service to create a .bif over the specified data and metadata file(s). The binding service utilizes enterprise services to authenticate and authorize the request. Next, the binder will use an enterprise retrieval service to gather the data and metadata from a storage reposi-

tory. Once the binding service generates the .bif, the service will store the .bif in a storage repository. Future repositories may exist for each element—one for data, one for metadata and one for .bif files—or in combination.

A validation requestor (shown in Figure 2) may be an access control mechanism or cross domain solution that is required to make a decision based on the contents of the data and metadata. The validation requestor submits a request to the validation service to verify the integrity and authenticity of the binding. Enterprise services will authenticate the validation requestor. Once authenticated, the request will be submitted to the validation service. The validation service will use retrieval services to gather the data, metadata, and .bif; verify that the files have not been maliciously or accidentally modified; and return the results (i.e., valid or invalid) to the requestor. Depending on the implementation environment, the binding and validation services could be deployed locally with all authentication and authorization checks occurring within a single community of interest (COI).

Future direction

Cryptographic binding is an enabling technology for systems that must rely on the integrity of data and metadata to make critical mission decisions including information dissemination and access control. The immediate goal is to make this capability operational by coordinating with the key figures in various pilots, experiments, and test environments within DoD, IC, allied/coalition, national, and international programs. These exercises will provide valuable feedback to improve this technology while allowing the capability to be used in controlled operational settings. In the near term, there are plans to conduct a security assessment of the ASN.1/CMS cryptographic binding proof of concept. The next steps are to complete a full, security-assessed refer-

ence implementation and standards profile for handoff to implementers. Cryptographic binding concepts and techniques need to be expanded and further proven to address evolving GIG net-centric environment needs including methods for high assurance bindings and envisioned security domains.

Open Source— Setting Software Free

The wireless age has thrown open the doors for easy access to communications, information, and even the software that runs behind the scenes. A generation of digital junkies has grown up expecting to get music and videos served up instantly and for free. Complete strangers exchange ideas and opinions through blogs, text messages, IMs, and tweets. The private details of our lives once closely guarded as too personal to reveal even to friends are routinely exposed on a host of public social networks. It comes as no surprise, then, that a grass-roots movement governing the open creation and distribution of software has a substantial following.

Open source software, with its minimal restrictions and costs, has been credited with presenting a formidable challenge to the traditional software licensing model. With major enterprises and even national governments mandating a shift to open source, what was once largely the domain of computer hobbyists may be evolving into the standard for programming in the future.

Proponents of open source software encourage the natural evolution of their creations through the collective efforts of experts and amateurs alike. Linus Torvalds, the inventor of Linux and de facto spokesperson for the open source community, credits individualism as the underlying characteristic of the movement's advocates. Just as the construction of Europe's great Gothic cathedrals depended on the personal talents of skilled craftsmen, modern-day "guilds" of programmers contribute their expertise as a kind of artistic form to create digital monuments. Open source coders generally take pride in adding a creative flourish, an innovative approach, or even a touch of whimsy that might be recognized by their peers as "a masterpiece." Unlike tightly controlled projects undertaken to enable market penetration or fulfill government contracts, open source software often emerges from the collective conscience of its ultimate users.

The libre years

Open source software can trace its roots deep in the heritage of the computer age. Collaboration among computing experts sharing a limited number of resources was essential for early software development. In the 1950s, the group SHARE provided IBM mainframe users a way to exchange technical information and develop a library of code that was available to its members. Until the late 1960s, after IBM unbundled its software for marketing flexibility, code was assumed to be *libre*—free for modification and redistribution.

Government-sponsored research continued to supply libre software long after proprietary products became the norm. The Advanced Research Projects Agency Network, ARPANET, relied on a strategy of offering libre software to foster global participation in the 1970s. This strategy eventually gave birth to the Internet, and it continues to fuel countless research efforts.

In 1984, GNU began supplying users with a free and open version of Unix-like software. The GNU organization promoted the freedom to copy, share, and change all versions of a software program. Any code developed as a GNU product continues to be subject to GNU's General Public License agreement, or GPL. A GPL, which is appropriately called a "copyleft" agreement, is intended to guarantee a user's rights to freely distribute free software and modify code in fee-based products. The following year, the Free Software Foundation (FSF) was formed as the holder of several GNU copyrights and served as the exclusive authority to enforce the GNU GPL. Later, running on a Linux kernel, GNU/Linux became the backbone of the FSF.

It wasn't until 1998 that *free software* was formally rechristened *open source software*, or OSS. The OSS community aggressively forged a path independent of closed-source software development strategies. Much of the success of OSS can be attributed to its cost—little to nothing. But a deeper commitment to open source, one that has grown out of an international community of developers and a philosophy of cooperation, may be what sustains it.

OSS taps into consumer markets

Businesses have been taking advantage of the efficiency—and low cost—of open source database management systems for several years. The open source database market is on track to surpass the billion-dollar mark by 2010, according to Forrester Research projections. Still, that is only a small chunk of an $18-billion market overall. MySQL is the clear leader in open source databases, accounting for about half of all installations for that segment. Operating under a GNU GPL, MySQL is the world's second largest independent open source company, trailing only Red Hat for top honors.

Because many open source database products are available at little or no cost, market share is more accurately measured by the number of installations instead of the amount of revenue generated. By this standard, 49 percent of the enterprises polled by Gartner Group in 2006 reported that they had deployed MySQL, compared with 67-percent and 61-percent deployments of SQL Server and Oracle, respectively. And MySQL was gaining market share at a 25-percent annual rate.

New to the open source database arena is a hybrid system unveiled in August 2009 by computer scientists at Yale University. HadoopDB was designed to incorporate the best of several successful technologies such as the approach taken by MapReduce, Google's software framework for conducting web searches, and parallel database management systems, which are particularly good at handling complex structured information such as tables containing trillions of rows of data. HadoopDB proponents claim the new architecture reduces the time it takes to perform computationally intensive tasks from days to hours. The database's developers foresee a role for HadoopDB in conducting analyses of complex systems such as stock markets, disease outbreak patterns, and climate change.

Open source software grabbed public attention with the introduction of the Linux operating system, in 1991. The adoption of Linux over the past 18 years has been gradual but steady. Led by the product's growing presence in enterprise servers and embedded software markets, open source operating systems have come to dominate the high-end computing market. According to the June 2009 ratings of supercomputers by TOP500.Org, more than 88.6 percent of the world's fastest computers were running some form of Linux, including all the top 10. But it was a low-cost alternative to home computing that vaulted OSS into the consumer market.

Netbook computers—subnotebook-size portables—have contributed significantly to the adoption of OSS. These low-end computers were designed to be affordable platforms for Internet browsing, Web 2.0 social interaction, and simple tasks like word processing or viewing photos.

Netbooks were thrust into the spotlight by the One Laptop Per Child (OLPC) project. To achieve its goal of providing affordable laptops for children everywhere, the OLPC foundation loaded its XO laptop with open source software for both the operating system and user software.

The 2007 launch of the XO prompted the introduction of several other computer brands that targeted the information technology needs of emerging markets. In addition to gaining a foothold in developing countries, the new netbook class of computers managed to establish a niche in mature markets. The netbook's low price made it a popular choice for entry-level computing, and OSS helped keep the cost down. Early on, nearly 90 percent of netbook computers ran on Linux products, but Microsoft rapidly overwhelmed the netbook market. The company reported boosting its share of netbooks in the US running on Windows from less than 10 percent in the first half 2008 to 96 percent by February 2009.

Netbook computers bridge the world of computers with a growing market of handheld products, another seemingly ideal environment for OSS. Consumer demand for smaller and smaller hardware has led to the rise in popularity of limited-function devices. The proliferation of mobile gadgets—Kindles, BlackBerries, TomToms, Droids—and the apps to customize their performance, has given OSS a boost that could lead to changing how software is developed in the future.

Open source operating systems typically integrate well with web-based services like Gmail, OpenOffice, and YouTube. As more and more services are being hosted online, the limitations of a lightweight open source operating system become irrelevant. Designers might justifiably ask, "Why add processing power to load native applications when all you need is a web browser that can pull more robust services from the cloud?"

Web applications running inside browsers and networked applications (netapps) have increasingly replaced the operating system as the dominate platform for building products and services. As long as web content is viewable with a common browser or netapp, consumers are generally indifferent to what operating system or software tool was used to create it. Users don't even need a computer to access Web 2.0 services. Any device that connects to the Internet—a cell phone, camera, GPS, music player, or even a digital photo frame—will suffice.

People around the globe are probably most familiar with open source software through the Internet. On July 31, 2009, Mozilla's Firefox logged its one-billionth download, less than five years after the open source browser was launched. Over 300 million users now surf the web using Firefox. Although Firefox still trails Microsoft Internet Explorer (IE) for web searches, its loyal and growing user base accounts for 31 percent of the Internet browser market. Mozilla's Asa Dotzler points out that if current trends continue, Firefox will overtake IE as early as January 2013.

Despite the anticipated growth of OSS, it is important to keep its adoption in perspective. While Microsoft's overall market share may be shrinking, most consumers and businesses still rely on Microsoft products—88 percent of computers in use today run a Microsoft developed operating system, while only one percent run an open source Linux product.

How secure is open source software?

The debate about the relative quality of open-source software over proprietary software has kept bloggers arguing for years. A five-year study by The Standish Group that was released in 2008 found that 70 percent of companies surveyed felt Red Hat Linux was less vulnerable to security attacks than Windows. But some contrarians propose that this perception is due to hackers mainly targeting Windows code, rather than fewer vulnerabilities in Linux. In a security review of open source products, Fortify Security Research Group determined most OSS lacks adequate documentation or even a secure development process. Security best practices were found to be a low priority for OSS developers, resulting in software plagued by numerous application vulnerabilities. A study conducted by computer security firm Secunia concluded the number of security bugs in Red Hat Linux exceeded the number of bugs in comparable Microsoft products. Many of the vulnerabilities in Red Hat were introduced through third-party components. The same study determined Firefox had considerably more security

bugs than Microsoft's Internet Explorer.

Open-source products still earn high marks for their quality. Software analysts at Coverity have been counting bugs in open source software for the Department of Homeland Security. Their findings in 2008 concluded code in 180 widely used open source software projects averaged 0.25 defects per 1,000 lines of code (KLOC)—one error for every 4,000 lines of code. This represents a 25 percent improvement over 2006 tests. One product had improved to the point that Covarity's test uncovered no defects at all. By comparison, Open Source Initiative president Michael Tiemann says proprietary software has consistently averaged 20 to 30 KLOC since the 1960s.

The future of open source

The world relies heavily on software from the United States, but some countries are looking to domestically produced open source solutions as a viable alternative. China has long been a global advocate for open source software. Many leading brands of computers in China are sold without an operating system preinstalled, giving consumers the option to add open source software. The high cost of proprietary software has fueled software piracy there, putting the country at odds with the global community. China's adoption of OSS is partly in response to software piracy, but open source products such as home-grown Red Flag Linux are also getting a boost as an expression of national pride.

Europe, like China, has also strongly embraced OSS. European-coded Ubuntu is a user-friendly version of Linux that is gaining market share globally, with Europe providing much of the operating system's support. As of summer 2008, Linux-based products were pre-installed on three percent of new computers in the UK.

The move to OSS can be seen globally through its adoption by various government agencies. In Southeast Asia, for example, the government of Vietnam issued a directive in early 2009 to convert all government servers, networks, and desktop applications to open source. As a hub for IT outsourcing, Vietnam views moving to OSS as a way to develop a local software industry.
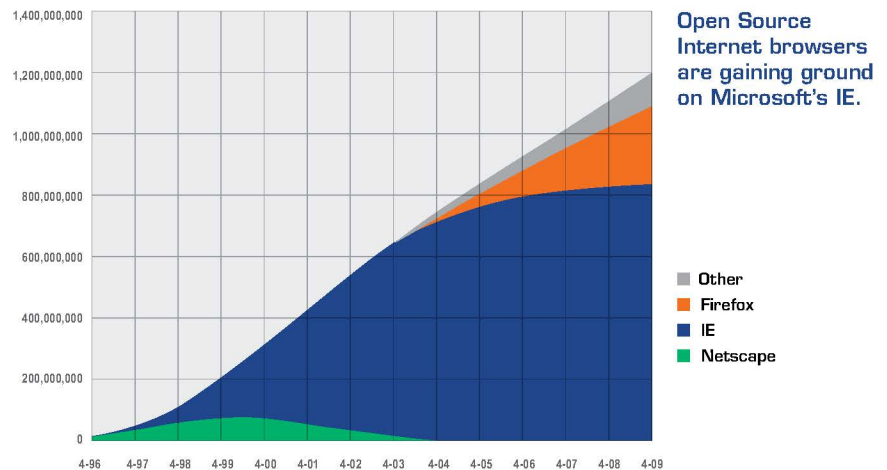
Industry leaders worldwide are conceding a growing need to support the OSS community, as well. Intel recently developed Moblin, a Linux-based operating system (OS) designed for the company's Atom x86 chip, to optimize Internet and

multimedia performance. The Atom chip is already found in many netbooks, and the anticipated proliferation of mobile Internet devices, or MIDs, should greatly expand its market penetration. Intel turned Moblin over to the Linux Foundation in April 2009. The Moblin.org group has recently rolled out the first beta of Moblin v2.0, which it expects to become the standard software development kit (SDK) for MIDs.

Google has also ventured into the open source domain as the company tries to gain a foothold in the software industry. Linux-based Chrome OS, set for release in late 2010, is designed primarily as a secure platform for Goggle's recently released Chrome browser. Building on the successful launch of Android, its open source OS and SDK for mobile devices, Google Chrome OS is targeted directly at the Microsoft juggernaut.

Even Microsoft supports a strategy to win over the OSS community in hopes of getting OSS vendors to port their software to Windows. Microsoft's Open Source Software Lab is working to integrate OSS with Microsoft Office, SQL Server database, and other Microsoft products. For customers who want to continue using Linux, Microsoft will offer Hyper-V, its forthcoming virtualization hypervisor.

For most consumers software is judged by what it can do rather than how it works. Such pragmatism will make it harder for closed-source software to compete with OSS solutions in the future. As the personal computer gives way to the mobile handset and services move to the cloud, open source software—whether it is used for the operating system, the web browser, or netapps—stands to gain market share and user acceptance.

Open Source Internet browsers are gaining ground on Microsoft's IE.

Legend: Other, Firefox, IE, Netscape

Credit: Asa Dotzler (Data from Net Applications)