Shutting out intruders with AppArmor

PROTECTIVE ARMOR

When an attacker succeeds in infecting a victim's system, the attacker inherits the victim's privileges. AppArmor beats the attack by reducing the potential victim's privileges to a minimum. BY RALF SPENNEBERG

Novell views AppArmor [1] as an easily configurable but effective protection system for Linux. According to the vendor, AppArmor competes with SE Linux, which has been part of the Suse distribution for quite a while now, although lacking the policies needed to run it. Whereas SE Linux is comparatively difficult to configure, but implements comprehensive MACs (Mandatory Access Control), AppArmor focuses on restricting the scope of individual applications.

The Task

It is an unfortunate fact that many programs suffer from bugs, and web applications are particularly badly hit. Most software is not coded by security specialists, though it may be publicly accessible via the web, and this makes it an easy target for attackers. If an attacker finds a programming error in an application, they can typically exploit the error, thus gaining access to the target system.

Even if the intruder only compromises a standard user account, this is still dangerous for the victim. The account gives the hacker direct access to locally installed tools. And a single vulnerability in a Set UID root program is all it takes for the attacker to take over the reins. Traditionally, admins and web masters have had no alternative but to keep their systems up-to-date and to remove any ballast, that is, to deploy only the software they really need. But none of this can protect you against zero day exploits – attacks that target previously unknown security holes.

How it Works

AppArmor is designed to help admins break out of the trap. The system monitors the way processes access files. AppArmor distinguishes between read and write access and also monitors the use of root privileges. Depending on your kernel, Linux can distinguish up to 29 capabilities (see *man 7 capabilities*). For example, *CAP_KILL* refers to root's ability to terminate processes, and *CAP_NET_RAW* allows root to create arbitrary network packages. The *ping* command needs this ability, for example.

The idea of controlling access and actions based on the program rather than the owner and/or user is not new. On the free BSD systems and Linux,

Immunix

AppArmor started out its career as a commercial product by Immunix, although it was known as Subdomain at the time. Novell acquired Immunix in mid 2005, renamed Subdomain to AppArmor, and licensed the code under the GPL early in 2006. Immunix was well known as a vendor of security solutions, most specifically due to the Stackguard compiler, a modified GCC that protects applications against variations on the theme of buffer overflow attacks.

Immunix was also heavily involved in the development of the LSM interface (Linux Security Modules) for kernel 2.6. Besides AppArmor, a number of security systems, such as LIDS (Linux Intrusion Detection System) and the competitor product SELinux, use the LSM interface to inject controls where they are needed in the kernel. Thanks to LSM, patches are not required, however, the LSM architecture is a subject of discussion for some projects.

Niels Provos' Systrace [7] implements this principle, for example. But while Systrace monitors system calls – as you would expect from the name – AppArmor uses the LSM Hooks (see the "Immunix" box).

Deploying AppArmor on Linux

Novell includes AppArmor with the commercial Suse Linux 10.0 and SLES 9 SP3 distributions (Suse Linux Enterprise Server 9, Service Pack 3). The GPL variant is included with Open Suse 10.1. You can install AppArmor on Open Suse 10.0, although this requires a time-consuming kernel patch. According to the mailing lists [2], you can expect Ubuntu and Fedora to support AppArmor in the future. This said, the GUI requires Yast 2 at this time of writing.

Novell has a number of packages up for grabs on Novell Forge [3]. The RPMs for the alpha version of Open Suse 10.1 will work on Open Suse 10.0. A kernel with AppArmor support is additionally required. The best approach is to use an original package from the kernel repository [4], for example, Linux 2.6.15, in combination with the *aa_2.0-2.6.15.patch* and *aa_namespace_sem-2.6.15.patch* [5] kernel patches. When you run *make oldconfig* to configure, you can typically just press [Enter] to accept the default values when prompted. The individual steps are shown in Listing 1.

In later patches, Novell will be renaming the kernel structure in Security FS to */sys/kernel/security/apparmor*. The Security FS has been part of the standard kernel since Linux 2.6.14.

Starting on Time

The userspace AppArmor component launches the system service and assigns a policy. The init script has kept the older Subdomain name: */etc/init.d/subdomain start* loads and enables the AppArmor kernel module. To allow the module to monitor an application, it has to be enabled before the application you want it to protect. This is why it makes sense to launch AppArmor at boot time. Also, the application needs a profile file below */etc/subdomain.d* (future distributions will use */etc/apparmor.d*).

Novell provides profiles for a whole bunch of critical commands, including servers such as Apache (in Prefork mode) and OpenSSH, for S-Bit tools

Listing 1: Setup Steps

```
tar xjvf linux-2.6.15.tar.bz2
cd linux-2.6.15
patch -p1<../aa_2.0-2.6.15.
   patch
patch -p1<..
   /aa_namespace_sem-2.6.15.
   patch
make oldconfig
make bzImage
make modules
make modules_install
make install
rmdir /subdomain
ln -s /sys/kernel/security
   /subdomain /subdomain
```

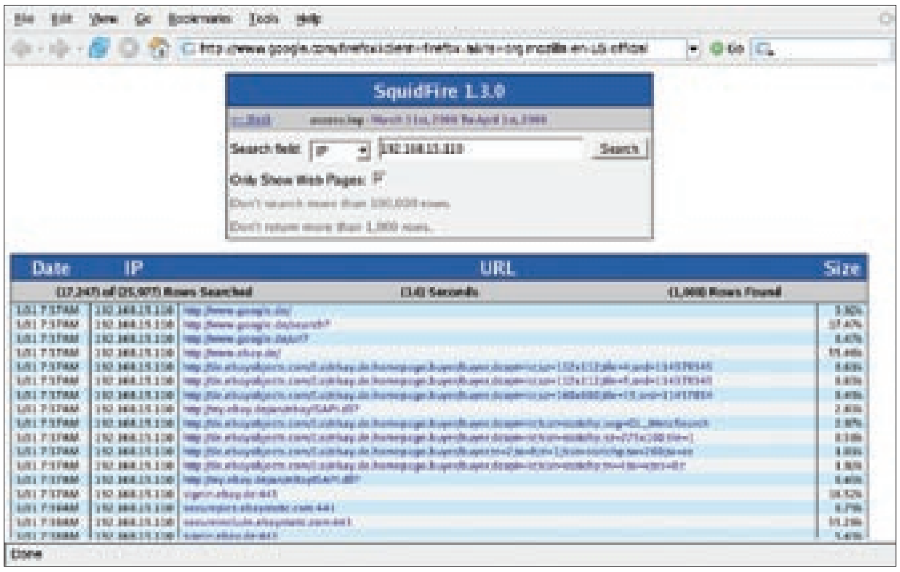


Figure 1: Squidfire parsing the access attempts logged by Squid. To restrict access to logfiles to this one web application, Apache runs mod_change_hat to enable AppArmor support. In this way, the security system knows which web application is currently active.

such as ping and man, network-capable clients such as Firefox and Real Player, viewers like Acrobat Reader, and even for the Klogd and Syslogd protocol services.

New Profiles

If you need to create additional profiles for your applications, the Yast-based profile wizard can help you set them up. The profile wizard just needs to know for which program you will be creating a profile at first. The user then launches the program, and runs the program in the normal way.

It is important to use all the application's functions at this stage. Make sure that attacks are impossible during the learning phase: AppArmor will later allow all the features that the application uses now. AppArmor learns the application's legitimate functions during this phase.

After quitting the application, the next step is to analyze the recorded events using the profile wizard. The wizard suggests an action in each case. If the monitored program calls another program, the profile wizard gives you the following choices: *Inherit* means that the same restrictions apply to the new application *kdialog* as to the application you are analyzing. *Profile* means that the program has its own profile. *Unconfined* means that AppArmor will not monitor the new process, and *Deny* stops the application from launching.

To facilitate the process of creating and managing profiles, AppArmor uses include files. The files are implemented as abstraction libraries and contain rules for standard legacy operations. For example, *#include <abstractions/kde>* allows access to the KDE configuration files and functions. Other profiles allow users to launch Bash or name resolution.

After successfully completing the wizard, it makes sense to relaunch the application and test how it performs under the watchful eye of AppArmor. If you notice that some functions are not working as advertised, you may need to rerun the wizard. In this case, the wizard reads the existing profile and updates the profile with your changes.

Listing 2 gives you a typical AppArmor profile for Kpdf. After the Vim comments (Suse provides a syntax highlighting module for Vim), the profile starts with the path to Kpdf; this specifies which program the policy governs. *flags=(complain)* switches the profile to complain mode, also known as learning mode. In this mode, AppArmor will warn you about infringements against the policy but without preventing the events from taking place. Toggling to *flags=(enforce)* tells AppArmor to restrict Kpdf's abilities.

Lines 4 through 10 reference a number of include files, and Lines 12 through 18 list the paths to which the PDF viewer is allowed access. An *r* following the path and file names refers to read access, whereas *rw* allows both read and write access.

Web Servers

AppArmor is particularly useful on web servers. In contrast to popular Mandatory Access Systems such as LIDS, GR Security, RSBAC, or SE Linux, AppArmor can monitor virtual hosts with different profiles on a web server. The Apache web server can change profiles depending on the current directory. Novell refers to this as the *change_hat* function – in what may be a humorous sideswipe at their competitor's red headgear.

But without some help from the Apache, AppArmor is not capable of ascertaining web server state. Novell provides a *mod_change_hat* module to handle this (the name will be changing to *mod_apparmor* in future). AppArmor allows programs to change their security context, however, the Apache web server is the only program to have implemented this feature as of this writing. An appli-

Listing 2: Kpdf Profile (Excerpt)

```
01 # vim:syntax=subdomain
02 # Last Modified: Sun Jan 22
   10:16:55 2006
03 /opt/kde3/bin/kpdf
   flags=(complain) {
04   #include <abstractions/
   authentication>
05   #include <abstractions/base>
06   #include <abstractions/bash>
07   #include <abstractions/
   gnome>
08   #include <abstractions/kde>
09   #include <abstractions/
   nameservice>
10   #include <abstractions/
   user-write>
11
12   / r,
13   /etc r,
14   /etc/X11/.kstylerc.lock rw,
15   /etc/X11/.qt_plugins_3.3rc.
   lock rw,
16   /etc/X11/.qtrc.lock rw,
17   /etc/exports r,
18   /etc/rpc r,
19 <I>[...]<I>
```

cation's main profile can have an arbitrary number of subprofiles (so-called hats) to support this. The hierarchy is restricted to one layer: subprofiles are not allowed to contain further subprofiles.

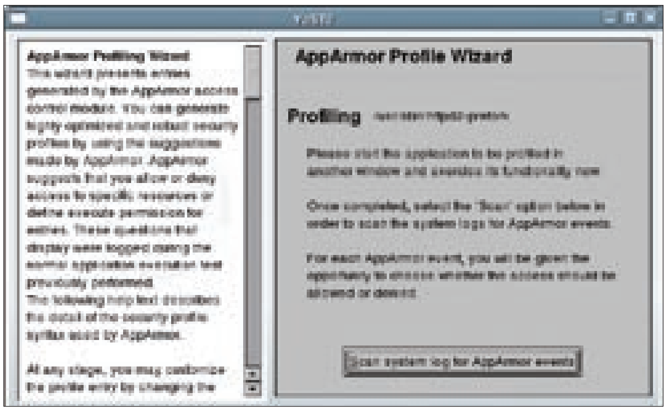
Changing Hats

Yast has GUI-based support for subprofile management. The command-line counterpart is more powerful, but Yast configuration is simpler. The following sections use the Squidfire web application (Figures 1 and [6]) to describe the Yast variant. Squidfire is a PHP script that makes Squid logfiles searchable. The AppArmor profile provided for this task, *usr.sbin.httpd2-prefork*, denies Apache, and thus Squidfire, all access to the logfiles, as the following */var/log/audit/audit.log* message confirms:

```
type=APPARMOR msg=audit↩
(1143872666.069:205): ↩
REJECTING r access to ↩
/var/log/squid (httpd2-prefork↩
(14820) profile /usr/sbin↩
/httpd2-prefork active ↩
DEFAULT_URI)
```

We need a subprofile to give Squidfire access. At the same time, this subprofile will restrict this access to the Squid logfiles only. This precaution will prevent cunning Squidfire users from running the evaluation tool against non-Squid files.

Again, the Yast profile wizard will handle the subprofile configuration. When you run the wizard, select Apache as the application. This tells AppArmor to allow all actions for this process and to log these actions for later analysis. After working with the application in your browser for a few minutes, click the *Scan system log for AppArmor events* button in the profile wizard (Figure 2) to complete the training phase. If you are training a change-hat-capable application, the profile wizard will suggest that you create a new subprofile (hat).


YaST

AppArmor Profiling Wizard
This wizard presents entries generated by the AppArmor access control module. You can generate highly optimized and robust security profiles by using the suggestions made by AppArmor. AppArmor suggests that you allow or deny access to specific resources or define execute permission for entries. These questions that display were logged during the normal application execution test previously performed.
The following help text describes the detail of the security profile syntax used by AppArmor.

At any stage, you may customize the profile entry by changing the

AppArmor Profile Wizard

Profiling /usr/sbin/httpd2-prefork

Please start the application to be profiled in another window and exercise its functionality now.

Once completed, select the "Scan" option below in order to scan the system logs for AppArmor events.

For each AppArmor event, you will be given the opportunity to choose whether the access should be allowed or denied.

Scan system log for AppArmor events

Figure 2: Click on Scan system log for AppArmor events, and the wizard will make a suggestion for the profile.

Strict Limits

Take care when responding to the profile wizard's prompts, particularly if the main application calls external programs. It makes sense to let these tools inherit the profile from the calling application.

When adding imates and CSS files, the Apache default profile is a sensible choice. After asking you a few questions, the profile wizard goes on to create a subprofile in *usr.sbin.httpd2-prefork* (Listing 3 shows an excerpt).

By default, the URI is used to distinguish between the various Apache subprofiles within the profile (see Line 2 of Listing 3). This example allows the */squid/index.php* web application to use Bash and read a number of system files. Listing 3 also uses the Squidfire components (Lines 11 through 15), and finally the listing evaluates the Squid and Apache access logs files (Lines 18 and 20).

On closer inspection, the subprofile actually emphasizes how dangerous some of the application's actions really are. The application obviously uses files with predictable names below */tmp* (Line 16), and it runs external Shell commands (See the reference to Bash in Line 6. You'll also find the *tail* command in Line 17.)

Distinguishing Directories

The *mod_change_hat* module lets you organize subprofiles for virtual hosts via the *Location*, and *Directory* directives. Administrators can tell the module which approach they prefer using the *ImmDefaultHatName* and *ImmHatName* directives.

The Imm prefix is still reminiscent of AppArmor's Immunix roots. Actually, the module has been renamed to *mod_apparmor* in later releases, and the keywords are now *AAHatName* and *AADefaultHatName*.

ImmDefaultHatName (or *AADefaultHatName*) selects a default subprofile for each virtual server. Additionally, subprofiles can be assigned to individual areas using the *Directory* or *Location* directives.

The following lines in the Apache configuration would thus assign a hat to the Squidfire web application:

```
<Directory "/srv/www/htdocs/squid">
  ImmHatName squidfire
</Directory>
```

You would need to call the subprofile *^squidfire* rather than *^squid/index.php* (Line 2 in Listing 3).

AppArmor gives system administrators a new approach to server security, especially in shared hosting environments where multiple customers share a web server.

Assigning a strict hat to each virtual host, and restricting the hat's access to files belonging to just one customer, would mitigate the danger of security holes in one customer's web application endangering other customers. This said, it would be a good idea to manually check the policies for this, or to create them manually from scratch, instead of relying on complain mode – but this should not be too difficult for most admins.

Conclusions

AppArmor locks critical applications away in a sandbox, restricting access to specific files, and limiting the system to executing specific commands.

If the application turns out to have a vulnerability that allows the attacker access to a shell or lets the attacker run commands with the victim's credentials, AppArmor steps in. The application is not allowed out of jail. Although the practical protection provided by a tool like AppArmor does not remove the vulnerability, the attacker will not be able to exploit the security hole to gain control of the system.

This principle protects a machine against the effect of zero-day exploits, and this makes AppArmor especially useful for programs that are accessible over networks, or programs that have to process data from untrusted sources (emails, images, video clips, office documents).

Changehat-aware applications can even handle state at runtime, applying different subprofiles as the situation dictates. This adds the ability to define profiles for specific web applications on web servers, applying specific rules based on the URI, virtual host, or directory path.

AppArmor offers a practical approach to Linux application security. The AppArmor alternative is an efficient option for users who don't want to contend with the additional complexity of security systems such as SELinux. For more on the relative merits of AppArmor and SELinux, see the AppArmor vs. SELinux comparison later in this issue. ■

Listing 3: Apache Subprofile

```
01 <I>[...]<I>
02 ^/squid/index.php {
03    #include <abstractions/bash>
04    #include <abstractions/nameservice>
05
06    /bin/bash ixr,
07    /dev/tty rw,
08    /etc/ld.so.cache r,
09    /lib/ld-2.3.90.so ixr,
10    /lib/lib*so* r,
11    /srv/www/htdocs/squid/cache.inc.php r,
12    /srv/www/htdocs/squid/config.inc.php r,
13    /srv/www/htdocs/squid/default_config.inc.php r,
14    /srv/www/htdocs/squid/index.php r,
15    /srv/www/htdocs/squid/parse_squid_row.inc.php r,
16    /tmp/access.log_1.3.0.inc r,
17    /usr/bin/tail ixr,
18    /var/log/apache2/access_log w,
19    /var/log/squid r,
20    /var/log/squid/access.log r,
21 }
```

INFO

[1] AppArmor: *http://www.opensuse.org/AppArmor*

[2] AppArmor mailing list: *http://forge.novell.com/mailman/listinfo/apparmor-general*

[3] AppArmor RPMs on Novell Forge: *http://forge.novell.com/modules/xfmod/project/?apparmor*

[4] Kernel repository: *http://www.kernel.org*

[5] AppArmor kernel patches from the January snapshot: *http://forgeftp.novell.com/apparmor/Development%20-%20January%20Snapshot/*

[6] Squidfire: *http://squidfire.sourceforge.net*

[7] Systrace: *http://www.systrace.org*