Security Patching Shared Libraries With KernelCare+

Linux shared libraries present serious vulnerabilities.

KernelCare

KernelCare

Security Patching Shared Libraries With KernelCare+

Linux shared libraries present serious vulnerabilities.

To operate web servers securely, it's not enough to patch their Linux kernels. Their shared software libraries must be patched as well. Otherwise, an enterprise leaves itself open to attacks that exploit vulnerabilities such as:

Heartbleed:

Attackers exploiting this OpenSSL request validation flaw could read a server's memory, then gain control of it. Immediately after it became known, it was used to steal the hospital records of 4.5 million patients. This vulnerability still exists on many systems, even though patches for it have been available since 2014

GHOST:

Attackers exploiting this glibc buffer overflow flaw could use gethostby* functions to make network requests that enabled them to gain control of a server. MySQL servers, Exim, and other mail servers were vulnerable to it. Once it became known, enterprises worldwide scrambled to patch it before it could be exploited.

OpenSSL and Glibc continue to present security problems on Linux systems. As of 2020,attacks on OpenSSL accounted for **71%** of vulnerabilities targeted in the technology industry. In 2020, Glibc was found to handle memory operations in a way that attackers could use to crash it and execute malicious code.

71%
of vulnerabilities targeted in the technology industry.

Vulnerabilities like this are why almost one in five attacks target OpenSSL, but it's not just OpenSSL and Glibc that put Linux servers at risk. Libarchive, a compression library included by default in a vast number of Linux distributions and software utilities, contains a vulnerability that can enable attackers to execute code on remote servers.

These sorts of library vulnerabilities are being discovered at an increasing rate: from 2017 to 2019, they nearly doubled in number. They're also becoming more widespread: In 2020, critical vulnerabilities known as Ripple20 were discovered in a widely-used TCP/IP library, exposing hundreds of millions of internet-connected devices to attack.

Patching libraries through server reboots is problematic.

The usual way that enterprises deal with library vulnerabilities is by rebooting their servers. Admins often don't know which services use which libraries, so they just reboot the whole server to update them all. These reboots, however, bring with them serious problems:

Server downtime: When servers are down, web sites go down, and display only error messages to visitors. After rebooting, it can take some time for server performance to stabilize, and occasionally servers don't come back up properly after a reboot.

Windows of vulnerability: Because rebooting is laborious and problematic, enterprises often only do it on a periodically scheduled basis, leaving their servers open to attack. Even if they reboot every 30 days to comply with security standards, their servers may be vulnerable for two weeks or more.

Even if they're patched manually, without a reboot, shared libraries may contain vulnerabilities. When libraries are updated on disk, old unpatched files can persist in a server's memory. What's more, vulnerability scanners don't detect these old unpatched library files in memory.

KernelCare+ patches shared libraries without rebooting.

Just like KernelCare, KernelCare+ patches the Linux kernel. It differs from KernelCare in that it patches libraries as well.

KernelCare+ patches shared libraries and detects library-related vulnerabilities. It even patches library files in memory, and does all this in a way that makes reboots unnecessary.

Right now, KernelCare+ patches the glibc and OpenSSL libraries, because these are the ones most often attacked. In the future, it will patch more shared libraries, such as those related to PHP and Python.

It employs new and sophisticated patching technology.

To patch shared libraries on web servers, KernelCare+ employs an innovative four-stage patching process:

1

The patch is created by the KernelCare team.

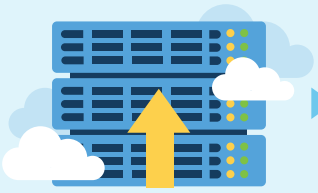
A library's source code--both original and patched--are translated into assembly language.

These files are compared, and the new patched code is put in a new section of the same ELF file.

After the code is compiled and linked, the patch is extracted from the resulting binaries.

The patch files are extracted from the ELF sections.

2



The patch is uploaded to the patch server.

The binary files are treated as a single patch, which is then uploaded to a dedicated KernelCare+ patch server.

The patch server then distributes the patch to customers' servers.

3

The patch is downloaded to the local agent

An agent program on each local server, lcarectl, "talks to" the patch server, which looks for known libraries on the local server.

The agent program then downloads the patch needed for each library present on the local server.

4

The patch is applied to the local server.

Using Linux APIs, memory near a library is allocated, and the patch is copied into it.

After ensuring that no threads are executing the old library code, the agent program reroutes calls from old code to the new patched versions via unconditional jumps.

Once this patching process is complete, the local server's libraries are fully protected against all known attacks.

See firsthand how KernelCare+ keeps servers safe.

Shared software libraries present serious security vulnerabilities that must be addressed. Many of these vulnerabilities must be addressed through patching, but traditional patching methods involve server reboots that present problems of their own. KernelCare+, in patching shared libraries without reboots, provides a better way to keep both kernels and libraries patched.

KernelCare+ employs new and sophisticated patching technology that addresses currentand emerging vulnerabilities in OpenSSL, glibc, and soon many other libraries as well.

To learn more about it and evaluate it in your environment with a free 30-day trial, visit: www.kernelcare.com