

# Libsafe: Transparent System-wide Protection Against Buffer Overflow Attacks

Timothy Tsai and Navjot Singh

Although security concerns encompass a wide range of issues and vulnerabilities, perhaps the most commonly exploited vulnerability is the buffer overflow vulnerability. Advisories issued by leading security organizations include many warnings about buffer overflow problems. Buffer overflows are significant in several ways. First, the most common outcome of a successful attack is a *root shell*, which is an interactive shell with root privileges. With a root shell, the attacker is able to modify or view files, install back doors, or set up further attacks, such as distributed denial of service attacks. Second, many buffer overflows can be triggered remotely by targeting commonly provided services. Since the attack is carried out at the application level, firewalls are useless in preventing attacks against required services. Third, the proliferation of exploit code permits even unsophisticated attackers to perpetrate buffer overflow attacks.

The main problem underlying buffer overflows is bad programming. A common data structure is the buffer or array. When the buffer contains characters, it is called a string. Consider the following example, which contains a buffer overflow vulnerability.

```
char *src = "hello world";
char dest[5];
strcpy(src, dest);
```

In this case, the `dest` buffer, which only has five bytes allocated to it, cannot fully contain the 11 bytes in the `src` string. However, because the `strcpy` function does not know the actual size of the `dest` buffer, it unwittingly copies the full 11 bytes of `src` to `dest`, overflowing the `dest` buffer and overwriting the contents of the memory immediately following `dest`. If `dest` exists on the stack, the overwritten memory may contain a stored return address. In such a case, when the return instruction corresponding to the overwritten return address is executed, the process control flow is diverted to the new return address. A common method of attack includes a short piece of attack code as part of the `src` string and a new return address that points to the start of the

attack code. Usually the attack code contains *shell code*, which simply spawns an interactive shell. If the attacked process executes with root privileges, as is the case with many system services, then the newly spawned shell also runs with root privileges.

The ostensibly perfect solution to buffer overflow vulnerabilities is to patch the code to eliminate the underlying programming bug. However, if the code has been widely deployed, then manual patching or upgrading involves significant logistical challenges. Furthermore, new latent buffer overflow bugs are constantly being discovered, as well as being created.

Libsafe is a run-time solution that inserts wrapper code at the start of functions that are deemed to be vulnerable to buffer overflows. A large number of known buffer overflow vulnerabilities are based on a relatively small set of standard library functions. The Libsafe wrapper code performs a sanity check on the buffers passed to the function as parameters. By intelligently examining the stack, the wrapper code estimates a maximum safe size for each destination buffer. If the size of the data written to the destination buffer does not exceed the maximum safe size, then no return addresses can be overwritten. Although this technique does not detect all buffer overflows, a surprisingly large number of known exploits are successfully detected.

The wrapper code for all functions is placed in a shared library. By leveraging a feature in the dynamic `ld.so` loader for Linux<sup>1</sup>, the Libsafe shared library is linked with all processes on the machine.

A key feature of Libsafe is its extreme ease of use. No source code or recompilation is required. Installation requires only a few minutes. No expertise regarding the machine or applications being protected, Libsafe, or even general security issues are needed. The performance overhead is minimal, usually around 1% vulnerabilities and exploits that have yet to be discovered.

Libsafe has been included in major Linux distributions. Direct download is also available from the project web site at <http://www.research.avayalabs.com/project/libsafe>. The web site also offers white papers, conference papers, and other documentation.

Avaya Labs Research,  
Basking Ridge, NJ USA,  
email: [ttsai@research.avayalabs.com](mailto:ttsai@research.avayalabs.com)

Avaya Labs Research,  
Basking Ridge, NJ USA,  
email: [singh@research.avayalabs.com](mailto:singh@research.avayalabs.com)

<sup>1</sup> Libsafe for Windows NT uses a different mechanism based on a registry key.