Buffer overflows are the ghosts that will always be among us

February 3 2015, by John Clark

The ghosts of Linux. Credit: BartM

Following the trend of giving catchy names to serious operating system security flaws, the Linux vulnerability revealed recently by security researchers Qualys has been called Ghost.

Like Heartbleed and Shellshock before it, the name is not plucked out of

the air but refers to the functions called "gethostbyname" in which the flaw appears.

These functions translate user-friendly domain addresses such as example.com into numerical network IP addresses, such as 93.184.216.34, and are part of the GNU C library which is included in practically every Linux system. This is important, as with most servers on the internet running Linux there are an enormous number of potentially vulnerable systems. Successfully exploited, the flaw could allowing an attacker to gain control of the system.

This is an example of a buffer overflow, one of the most persistent types of security problems that appears endlessly in lists of security vulnerabilities. For any computer security researcher it's a case of déjà vu.

Ghost hunting

A function is assigned a certain amount of memory allocation to store the parameters or data it uses. A buffer overflow attack works because the function doesn't correctly define or check the parameters it is sent. A malicious user can supply parameters larger than the allocated memory space which results in them being written into memory space outside that allocated – and therefore beyond whatever security restrictions had been placed on it. If this data is executable code, the system can be fooled into running it, potentially with greater system privileges.

The amount of memory that can be overwritten in the Ghost vulnerability is really very small (either four or eight bytes, depending on whether the system is 32-bit or 64-bit). But even this tiny amount of memory may be sufficient to allow a complete compromise of the system. The degree of skill needed to exploit this particular bug may be very high but Qualys has offered an example of code that exploits the

flaw based on something as simple as sending an email to a mail server.

Very few applications are known to be remotely exploitable – and many more recent applications don't use the gethostname functions at all. However, applications using the PHP coding language are a significant source of concern – for example, the popular WordPress blogging software is identified as potentially susceptible, so it's not just obscure software that's affected.

Buffer overflows are part of an even larger collection of exploits arising due to lack of proper parameter checking. In many online database access applications, a malicious user (or application) can supply input parameters that have been specially crafted so that they override any built-in checking. The most common of these is known as an SQL injection attack. Buffer overflows and SQL injection attacks are similar in that both exploit deliberately malformed data sent to program functions that cannot properly process it, and both exploit the absence of proper checking.

This is largely an avoidable problem. There have been concerted efforts by the software development world to seek out and fix buffer overflows in code. It seems, however, that they will always be with us.

Write once, check twice

Qualys have worked with various Linux distributors in advance of announcing the vulnerability so that patches for all major distributions have been available since January 27, 2015. If you are running a variant of Linux such as Debian 7, Red Hat Enterprise Linux 6/7, CentOS 6/7, and Ubuntu 12.04, you would do very well to ensure that your system patches are up to date.

Taking a step back, the reaction of the computing community will be a

mixture of "yikes", "phew", and "yawn". The first, because the vulnerability is present in a significant number of systems worldwide. The second, because in a great many cases it's difficult to exploit and so there's time to roll out the patches that fix the problem. And the third, because we've seen it all before.

This particular flaw was recognised and fixed as far back as 2013 – and may have been present since around 2000. However as the fix was not classified as a security problem many popular distributions of Linux didn't include it in updates.

And so it comes back to haunt us – and it will certainly not be the last of such vulnerabilities we see. To make buffer overflows a thing of the past will require an enormous amount of due diligence – systematic, thorough code review and testing – as new code is written. But the sheer volume of code that exists, such as the potentially 15-year-old lines that include this flaw, never mind that being written anew, should give some indication of the scale of the task.

Source: The Conversation