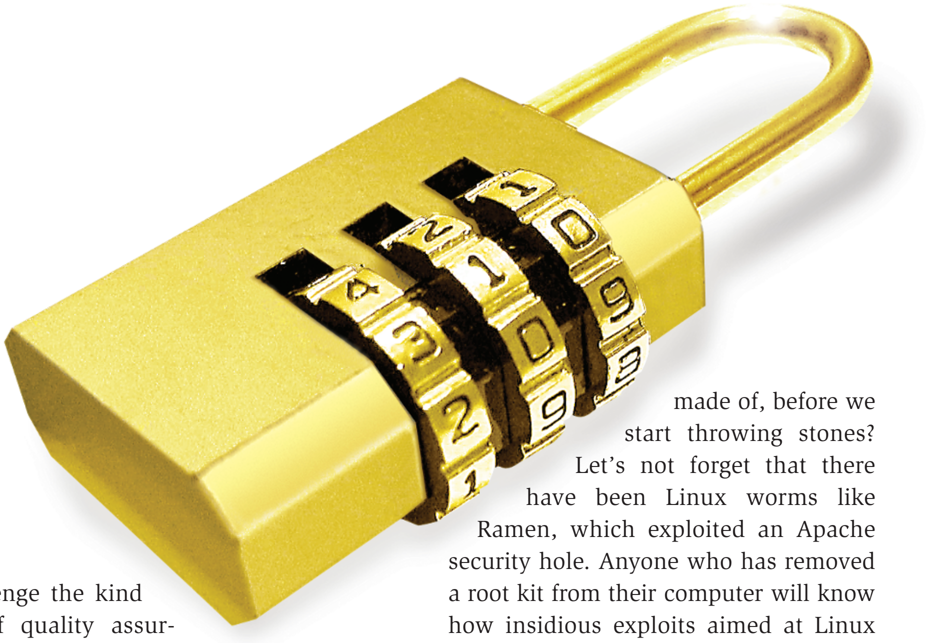
Open Source and Security – An Inside View

Open Source – Is it safe?

Enemies of Open Source claim that it is the root of all security holes. Advocates of free software maintain that the opposite is true. Who is right? Neither camp, or possibly both? This article on Open Source and security provides the answers.

BY OLAF KIRCH



One of the oldest clichés about Linux development is the notion that the Kernel, and whatever else might happen to be running on a Linux box, is the product of a more or less spontaneous development process. Like a movie of an explosion running in reverse. Thousands of individual components picked up by the maelstrom, thrown together, and suddenly gelling to form a whole – with a satisfying thud from the sound effects department thrown in to boot.

Many outsiders may perceive the development process of free software in this way. Also, many Linux developers would not contradict this cliché, as the image of spontaneous, self-organization reflects the kind of grassroots idealism that keeps most of us going, despite the trend toward commercialization.

But no matter what relation this image actually bears to real life, I do not intend to write yet another Open Source manifesto – there are enough of them around as it is. Instead, let's stick to the facts, using software security as a measure.

Quality Assurance and Open Source?

People who regard the genesis of Open Source software as a kind of global hacker happening, will of course challenge the kind of quality assurance that a process of this kind can provide, not to mention the security criteria.

“Of course Linux is secure!”, most Linux fans will assure them. But what justification is there for this statement? Or, to put that a different way, how can security be quantified? And: Does Linux development really comprise a security process? And: How can we ensure strong security standards in future?

Is Security Quantifiable?

It is no secret that Microsoft continues to face credibility issues with respect to software security and Trustworthy Computing. And every six months or so, some bored script kiddie paralyzes half the civilized world with a more or less incompetently coded worm.

One should be wary of accepting the figures on the financial damage supposedly caused by a worm at face value, but even if you delete the last zero from half-a-billion US dollars (the estimated damage caused by the ILoveYou virus), that still adds up to a lot of money.

But poking fun at Microsoft is cheap. Admittedly, MS did not take security as seriously as it should have, but shouldn't we check to see what our own house is made of, before we start throwing stones? Let's not forget that there have been Linux worms like Ramen, which exploited an Apache security hole. Anyone who has removed a root kit from their computer will know how insidious exploits aimed at Linux bugs can be.

Many of these exploits are tailor-made for script kiddies. Just download the tar archive, extract the files, and run the script to check whole blocks of IP addresses for vulnerable services and install backdoors on any victims it finds.

Attacks of this kind are accompanied by increasingly sneaky techniques to hide backdoors from admins: from modified system programs such as *ps*, *top*, and *ls* to loadable kernel modules like *adore* that hide complete processes and files. From here, it is a small step to a self-propagating worm.

Does this mean that Linux is just as (in)secure as Windows? Which leads us directly to the issue of whether security is quantifiable, and if so, how?

The CERT Vulnerability Database

The CERT Vulnerability Database makes for interesting browsing. CERT stores any incidents that comes to its attention in a repository. These can be security holes in operating systems, virus reports, or router vulnerabilities.

Depending on the type and severity, CERT may release vulnerability notes, incident notes, or advisories on these occurrences, where an advisory reflects a more serious scenario.

The database (or at least the publicly accessible part of it) also provides a numerical rating for each vulnerability as a general indication of its severity. The rating reflects several aspects such as how widespread the problem is, if it threatens the infrastructure of the Internet, and other parameters.

The vulnerability notes of the last few years can be roughly categorized by operating system and plotted as shown in Figure 1.

But let's not jump to premature conclusions just from looking at this graph. The figures are not a direct measure of the security of an operating system. Instead, they simply tell you how many security holes became public knowledge, and indicate how serious CERT thinks they are. The bars provide a (more or less accurate) indication of how endangered each type of system is.

The graph only indicates the vulnerability of Linux-specific components under the the "Linux" label. This means the kernel or low-level services that support the kernel. Other applications, such as Apache or OpenSSH, are grouped below "Open Source". The Unix category contains only Unix-specific incidents, such as the holes in various CDE services. The "3rd Party" category comprises mainly Windows applications, such as FTP servers and mail clients programmed by third parties rather than by Microsoft.

The graph clearly indicates a number of trends – most notably, exponential growth. It does not really matter whether one is 50-times more likely to be hacked today, in comparison with 1996. But the trend is clear, even if one ignores the absolute values, and this explains why security has become such an important topic in recent years.

The second thing that becomes apparent is the fact that vulnerabilities are on the increase in more or less every category. Of course, some of them show more pronounced growth – the Microsoft bar has doubled its size every year, for example – and some less (Linux + Open Source). The Unix category even dropped slightly following a particularly "fat" year in 2001, when a whole bunch of CDE issues was discovered.

Does this allow us to conclude that Linux is inherently more secure than Windows? No. As previously mentioned, the figures show that CERT considers Microsoft systems to be more vulnerable than Linux systems. Why this should be so, is purely speculative. The security of a system as a whole not only depends on the quality of the software, but also on a number of other aspects, such as the level of security provided by the default configuration of its components, or the knowledge-level of the average user.

So what makes Linux secure?

If you take the trouble to read the technical details on Microsoft Security Updates, you are sure to meet some old acquaintances: kernel bugs, buffer overflows in Web server modules, in the SQL server – but Linux has these too. And this is hardly surprising. Despite all the differences in the philosophy, there are some major similarities. For example, a monolithic kernel, a privilege model based on user IDs, an administrative superuser equipped with a full set of privileges.

One could argue that major bugs occur less frequently under Linux, or that they are published less frequently. At least this is what the CERT figures would seem to indicate.

What makes this all the more remarkable is the fact that the Open Source development model makes it easier for attackers to discover holes. On the one hand you can search the source code for vulnerabilities, using all kinds of tools (from simple greps to a line-by-line audit). And this is an option that many people take – not only the "Black Hats", but also those with an interest in making Linux more secure. Security consultants and agencies, and of course, Linux distributors like Red Hat and Suse, perform regular audits of components relevant to the security of Linux.

On the other hand there is the development process itself, which (with few exceptions) is an open process that makes it more or less impossible to brush issues under the carpet. There is a long history of developer groups who have tried to solve security issues in their projects clandestinely, and have failed. Sudden, unannounced version releases make malevolent hackers sit up and listen. There are even people who monitor the patches produced by some projects and recognize a fix for a buffer overflow, no matter how innocuous the log entries or release notes may be.

In other words, Open Source software is subject to the kind of quality assurance that can be painful to the developer, but in the end will produce a high standard of software quality. But that does not mean there is no room for improvement, as the OpenBSD team is only too pleased to demonstrate.

Another question that has to be looked at is the extent to which a disclosed security hole can be exploited on a typical system. Many years ago Linux distributors started adopting the security strategy of reducing the number of services enabled by default to a minimum. Now Microsoft is following suit, and making a big fuss about it, by launching its Trust-

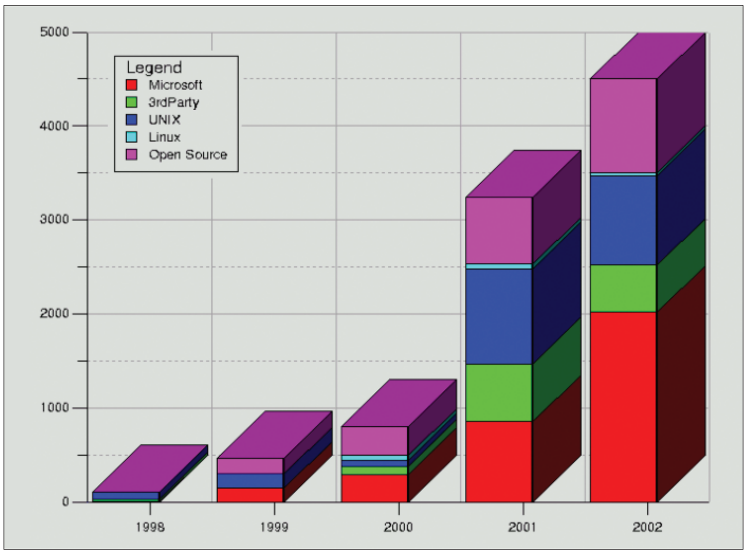


Figure 1: The CERT Vulnerability Notes statistic provides an overview of vulnerable systems. Note the exponential growth overall

worthy Computing campaign. And it is to be hoped that no-brain features such as the amazingly naive handling of attachments in Outlook are something that we will never see on Linux.

Certified Security

That may sound fine, and may be enough to convince some virus-harassed users to switch to Linux. But for others it is not enough – they demand more concrete evidence.

Rightly so. We have become accustomed to independent authorities checking the security and functionality of our surroundings. After all, would you use an automobile that the boy next door had duct-taped together in your neighbor's garage on a public highway? Like, "My friends all used it and nothing happened", well big deal, but no thanks anyway.

But come to think of it, is this not exactly what IT manufacturers demand of their users, although the bodywork may be shiny and the seat covers have a nice floral pattern? Linux development is just a prominent example of exactly that.

There is no mandatory security check for software, although there are some efforts towards introducing one, such as the US authorities insisting, in the last two years, that manufacturers of operating systems meet the Common Criteria requirements – a stance that European states are currently thinking of adopting. Similarly, many large enterprises insist on certification for any operating system that they intend to use.

This is why Linux distributors are committed to CC evaluation. In February last year Red Hat and Oracle released a joint-announcement stating that they were working on EAL2 certification. In August last year Suse and IBM went through EAL2+ evaluation, and EAL3+ is in the pipeline.

The whole thing sounds a lot like a happening for paper tigers. And it will not surprise many readers to hear that certification is mainly a question of getting the paperwork right. Having said that, one should not simply dismiss certification by equating the lack of new code with an accompanying lack of new security features. In fact, the opposite is true. Documenting processes that influence a product's security stance is a major part of the certification process. For example, how a product is tested, how security updates are handled, or how to ensure that the traceroute version that makes its way into the final release not only works properly, but also does not include a Trojan smuggled into the developer CVS by some intruder.

Within this framework, certifications can provide the kind of transparency that can help a product shake off its college dorm stigma.

Trusted Computing?

The "Trusted Computing Group" is surrounded by techno-hype. Some maintain that it is the greatest innovation since the invention of the lock, others see it as the final curtain for open software.

If you ask me, it is neither. The TPM chip, and any technologies based on it, such as Intel's LaGrande, can enhance the security of an operating system. This said, it provides no protection against buffer overflows, or any other kind of bug. Also, you can completely ignore TPM – you can compile a Linux kernel and run it on a TPM/LaGrande motherboard just like on any other.

The question is, should we ignore this technology, or does it actually provide something that can help make Linux more secure? That is something worth talking about, although the hardware manufacturers are doing very little to facilitate or encourage this kind of discussion at the present time.

On the downside, there is a danger of TPM becoming an obstacle to interoperability. For example, CIFS clients could refuse to talk to a Domain Controller without Microsoft certification. Scenarios of this kind may be alarming, but they are certainly not restricted to a single technology, such as TPM.

New Approaches

If we look at this situation from a different point of view, the discussion is as follows: Linux may have similar architectural problems to Windows, but it has a head start in dealing with them. That is, Linux can justifiably claim to be the secure alternative to Windows. The community cannot afford to rest on its laurels, however. Microsoft has pulled out to overtake, and it doesn't make sense to sit tight and let this happen.

In my humble opinion, we need to think about new approaches. The current approach of continually searching for, and fixing, security holes, is adequate, but it does not scale well to meet steadily increasing security requirements.

Designs such as SELinux are promising, but simply adding more security facilities to the kernel is not enough. Applications need to leverage these facilities, and that means a lot more work. It is difficult to convince software developers that their applications, no matter how sexy they may be, are a security hazard, as they have to be installed *setuid root*. Many authors simply retort by saying, "Show me the buffer overflow, and I'll admit I have a problem." But the opposite tack is also difficult, "Write your application so as to demonstrate that the section requiring increased privileges is secure".

Thinking about designs that can leverage TPM and LaGrande will not change anything. The kind of segregation that LaGrande envisages is more strict than any current design as provided by privilege separation and chroot jails.

The process is going to hurt, and the results will not be satisfying all the time, but stagnation would be a lot worse. ■

INFO

[1] CERT Vulnerability Database: *http://www.kb.cert.org/vuls*

[2] Common Criteria Certification: *http://www.commoncriteria.org/*

[3] Red Hat/Oracle CC Certification: *http://www.redhat.com/solutions/industries/government/commoncriteria/*

[4] Suse CC Certification: *http://www.suse.com/us/company/press/press_releases/archive03/security_certification.html*

[5] "Trusted Computing: *https://www.trustedcomputinggroup.org*

[6] Info on LaGrande: *http://www.extremetech.com/print_article/0,3998,a=107418,00.asp*

THE AUTHOR

Olaf Kirch has been an active Linux user and supporter for over ten years and currently works for Suse, where he is involved in IPv6, NFS, and security.