news@UK

The Newsletter of UKUUG, the UK's Unix and Open Systems Users Group
Published electronically at `http://www.ukuug.org/newsletter/`

Volume 17, Number 1 **ISSN 0965-9412** **March 2008**

Contents

News from the Secretariat

Jane Morrison

Thank you to everyone who has kindly sent in their subscription payments so promptly. We have received a number of early payments. Those remaining outstanding will be chased this month and any not paid at the end of April will not receive the next issue (June) Newsletter.

At the time of writing the two day February seminar (FAB 2008) on the topic “Files and Backups” has just taken place. Both days were very well attended and I understand the event was enjoyed by all. A write-up of the event is included in this newsletter. The speaker papers from FAB 2008 can be found on our web site at:
`http://www.ukuug.org/events/seminars/fab`

We now have everything in place for the UKUUG Spring Conference and Tutorials, being held in Birmingham (31st March - 2nd April). Once again Google have very kindly provided sponsorship which has enabled us to organise the Conference Dinner free of charge for all Conference delegates. Other sponsors include UKUUG Gold sponsor members Novell and Sun Microsystems and event sponsors Transitive and Bytemark. The conference web pages include full details of the event and also a booking system:
`http://spring2008.ukuug.org/`

We expect to organise a Linux event as usual this year, which will probably take place in early Autumn. Details will be sent out as soon as definite dates are set.

The next Newsletter will be the June issue and the copy date is: Wednesday 21st May. Articles and comments, please, to
`newsletter@ukuug.org`

Chairman’s report

Alain Williams

I write this shortly after the successful FAB 2008 seminars: the talks were interesting as were the discussions in the bar afterwards. It is events such as this, the bringing together of people with a common interest, that makes the UKUUG useful to its members. Many of us are solving similar problems, the exchange of ideas is invaluable.

I expect that our Spring 2008 conference at the Birmingham Conservatoire (31st March to 2nd April) will be equally successful – it had better be, a lot of hard work has gone in to organising it, thank you John and Zeth. Book your place now.

At Birmingham will be the first event of the new UKUUG PostgreSQL SIG (Special Interest Group), we have as organiser Simon Riggs who is enthusiastic about PostgreSQL.

An innovation at Spring 2008 will be BoF (Bird of a Feather) sessions. These are where people with a shared interest can get together to talk. By getting together with other specialists you can solve your problems more effectively that you can by yourself. You can plan your BoF during the Conference or, even better, beforehand on the conference Wiki:
`http://spring2008.ukuug.org/community`

UKUUG is keen to support this sort of activity; we are good at the organisational aspects – we just need people who are active in a field and will put in a bit of time working with others and then we can create another SIG – much to everyone’s benefit.

Worried that it might not fly? Go to the conference Wiki, suggest a topic, see what happens. Feel free to contact me for a chat.

With regret I have to announce that Mike Banahan has left UKUUG Council since his business is growing and demanding more time. Thank you Mike for what you did while on Council.

Summary: UKUUG exists to serve its members, please help us to provide what you want by telling us what we should be doing, even better work with us to bring people in your sphere if interest together.

Lug Radio Live announcement

The LUGRadio live conference will be held on the 19th-20th July at Wolverhampton University Students Union.

The conference, which has grown out of the highly succesfull LUGRadio podcast, is now in its fourth year has established itself as the UKs premier community conference with its avowed informality and an eclectic mix of talks, exhibitors and beer.

While no speaker list is available speakers from previous years include Mark Shuttleworth, Alan Cox, Micheal Meeks and many others all for an entry fee of £5 for the weekend.

It probably wont be the most technical conference you go to this year, indeed it tries hard to cater for newcomers to linux as much the hardcore geek but it should be the most fun.

Files and Backup Seminars – London

Roger Whittaker

On the 19th and 20th of February the Files and Backup Seminars (FAB 2008) were held at the Imperial Hotel in Russell Square, London. This was one of a series of one and two day meetings which UKUUG are presenting from time to time.

The event was a success with more than 70 attending, and a variety of interesting talks were presented.

After an introduction from Alain Williams on behalf of UKUUG, the first day opened with a talk from Chris Proctor of Open Minds. Chris spoke fairly generally about the principles of High Availability: most of what he said was equally applicable to Heartbeat 2 or to proprietary solutions such as Steeleye's Lifekeeper. He emphasised the fact that the aim of this approach is to eliminate single points of failure, and that the purpose is to keep the application running whatever happens at the lower levels. He discussed different approaches to the dreaded split-brain problem and the use of STONITH (the wonderful acronym for "shoot the other node in the head"). He also mentioned the use of high availability as a disaster recovery solution across geographically separated sites, and DRBD for keeping separated disks in sync.

Simon Wilkinson of the University of Edinburgh then spoke on OpenAFS. He touched on the history of OpenAFS (it became Free Software in 2000: had it been Free for longer it could perhaps be much more widely used now). He went on to explain the basic principles of operation of this secure federated network filesystem, and explained the concept of a "cell" and the fact that global access is possible across cells and that you can mount public OpenAFS filesystems across the internet. He discussed some very large implementations (both commercial and academic) and looked to the future with disconnected operation a not too distant prospect.

Steve Whitehouse of Red Hat then spoke on GFS2. GFS2 is a symmetric cluster filesystem which relies on Red Hat's cluster tool cman for its operation. This was mainly a discussion at the level of the coding work rather than the usage of the filesystem, but he encouraged the audience to test the version currently in Fedora Rawhide.

Jarod Nash of Sun Microsystems created a lot of interest with his talk entitled "ZFS Under the Hood" in which he explained the design and operation of the revolutionary ZFS file system, which combines the concepts of file system and volume management, and uses the copy-on-write principle and checksums to ensure that data and metadata on disk are always consistent. He concluded his talk with a demonstration of the basic commands, creating filesystems using a pool of two external disks connected to his laptop.

Chris Proctor spoke again, this time mostly off the cuff, with some interesting and amusing stories of from the support side illustrating how using sophisticated solutions designed to preserve your services and data at all costs only works if you know what you are trying to achieve rather than just seeing them as an insurance policy that obviates the need to think. Some of his support horror stories struck a chord with many in the audience.

The first day ended with a talk by Dr Charles Curran of CERN (no relation as far as I know to the former UKUUG Chair). He spoke about the difficulties of managing and storing the vast amounts of data generated by particle physics experiments with limited resources in terms of hardware, and with projected storage requirements outstripping expected growth in funding. His talk was by turns informative, amusing, sometimes indiscreet and very much enjoyed by all present.

The second day started with Howard Thomson of UKUUG Council and Gem Brain. His talk was entitled "Backup integration with filesystem snapshot capability". He started by discussing the use of LVM snapshots on Linux and "shadow copy" on Windows as a means of performing backups. He pointed out some of the problems caused by lack of integration between LVM and the filesystem, meaning that an LVM snapshot on a running system did not always necessarily provide a fully consistent view of the system. More generally he noted that the lack of a holistic approach to the entire system in the Open Source world made it more difficult to get the kind of integration between the behaviour of applications, filesystems and volumes needed to get this kind of thing right. He pleaded for concepts from the world of databases to be applied to filesystems and in particular the idea of large transactions. For example, he suggested that the entire set of filesystem writes involved in installing a package should ideally be a single filesystem transaction. He also mentioned some interesting new developments in the world of filesystems, including Btrfs, HammerFS, CRFS and POHMELFS. This was a thought-provoking talk which raised some fundamental points about how things (don't) work and how they perhaps should.

Ben Harris of Cambridge University spoke on "Reliable backups without tapes", describing a disk based backup system which he manages there. He explained the evolution of the system from a previous tape-based system, and the many iterations he went through to be sure of the security and reliability of the stored data.

Kern Sibbald then presented on the Bacula project which he initiated eight years ago and still leads. He explained the design goals and principles of this popular open source backup system and showed a demonstration in which he backed up and restored a MySQL database. Some very large sites are using Bacula: this impressive talk suggested that they are right to trust it.

The final main session was given jointly by Adam Spiers and Jo de Baer of Novell, with the title "Open Storage Management". This session examined emerging technologies in the areas of virtualisation, server management, storage and virtualised storage. In a stimulating but acronym-laden talk they looked at the use of CIM (the Common Information Model) and WBEM (Web Based Enterprise Management) for managing servers and virtual machines, storage management using the Aperi project, virtual fabrics (VSANs by analogy with VLANs) and NPIV (N_Port ID Virtualization) as well as Novell's ZOS (Zenworks Orchestrator). They showed some live demonstrations, including starting up a virtual machine using WBEM and managing virtual machines using ZOS.

After the programmed sessions were over, there was time for some lightning talks by members of the audience. Damien Brasher of Southampton University spoke about his proposed Distributed Internet Archive Protocol:
http://www.diap.org.uk/

Nick Mailer of Positive Internet began a highly articulate rant on the general theme of "don't trust black box solutions", referring to hardware raid controllers, and concluding that "the layer that's supposed to protect you comes back and bites you". He then changed tack and pleaded for a usable solution to the problem of recording and logging which files have been changed in real time so that backup tools do not need to check all files. A member of the audience suggested a possible solution based on systemtap.

Carl Elkins of Genome Research shared some of the problems of backing up 144 TB of data over a weekend.

Alain Williams outlined and recommended a backup system that he offers to customers based on rsync with the `--link-dest` hard linking option switched on. There was some discussion about the scalability of such a system over time.

Finally Keith Matthews of Frequentious Consulting discussed the problems of looking after data for graphic designers who require 10GB of disk space per month per designer that needs to be archived for years into the future. He also recommended OpenFiler (from xinit Systems).

On both days there was a good quality buffet lunch with plenty of choices, and coffee and biscuits at appropriate times during the day. A nice additional feature was that there was a drink at the bar (*free as in beer*) "on the house" at the end of each day.

I enjoyed the event very much, and in talking to others between sessions I got the impression that this feeling was general.

UKUUG plans to offer more sessions of this type: any suggestions about suitable topics, venues or speakers are always welcome.

Microsoft and Yahoo! – Oh my!

Peter H Salus

Just before Groundhog Day, Microsoft offered US $44.6 billion to Yahoo! in a hostile buy out. An obvious attempt at a strike against Google. Google has over 66% of Web search business; Yahoo! has just 20%; Microsoft, 7%; and Ask, 4% according to C|Net News (8 January 2008).

Scott Rosenberg, one of the founders of Salon, wrote that he saw this as "a path to failure for both companies". I happen to agree, but I think it's worth examining the entire affair.

Microsoft is the epitome of "old" capitalism. Over the past 25 years it has grown by engulfing and emulating. It has always been late on the adoption of the new. The Internet was "irrelevant". Games and music were add-ons. Search engines were an afterthought.

Just look at the decade's "major acquisitions".

- WebTV (1997)
- Hotmail (1998)
- Visio (1999)
- Great Plains (2001)
- Navision (2002)
- Tellme (2007)
- aQuantive (2007)
- Fast (2008)

They totalled about $12.5 billion.

While Microsoft likes to point to its track record as a reason why it will be able to handle integration issues where Yahoo! is concerned, it's important to note that nearly every one of these acquisitions took Microsoft into new businesses, with minimal overlap. And most recently came this offer for Yahoo! – the last "old media" company. A purchase of Yahoo! will have tons of overlap, not to mention all of the cultural issues.

Yahoo! owns or controls content, it markets to bring in audience and floods them with ads. Google arrives on my desktop with content, ads and tools. Yahoo! is centralized; Google is distributed. Microsoft, too, operates on an old control model – as I wrote above, it is the last of the "old" technology companies – controlled, closed . . . never open, never offering choice.

Think back. UNIX from AT&T originally ran only on DEC PDP machines. Then came the proliferation of hardware and the variety of UNIX systems. One of the big criticisms was the fact that there were so many "incompatible" UNIXes.

Then came Linux. And Linux runs on nearly every box, nearly every kind of chip. But (oh dear!) there are so many different kinds – over 350 distributions. You're not compelled to run Vista. You can choose. You can opt.

Nicholas Carr, former editor of The Harvard Business Review, said "Microsoft makes its money selling licenses to millions and millions of people who install it on individual hard drives. Most of what you need is on the Internet – and it's free. There are early warning signs that the traditional Microsoft programs are losing their grip".

On February 3, John Markoff wrote in the New York Times: "In moving to buy Yahoo, Microsoft may be firing the final shot of yesterday's war. ...Silicon Valley favors bottom-up innovation instead of growth by acquisition. The region's investment money and brain power are tuned to start-ups that can anticipate the next big thing rather than chase the last one".

Dave Winer (of scripting.com) wrote: "Does Yahoo + Microsoft make sense? Nahh. It's like the dead leading the blind. The only reason the deal makes sense is because it's the only thing either company could do that anyone might possibly care about".

I mentioned culture earlier. My guess is that in any merger, a large number of Yahoo! employees would jump ship. In my experience, it isn't the average software engineer or graphics designer or systems person who leaves. There a lots of jobs in Silicon Valley. In a first-rate article (New York Times, February 4), Gary Rivlin and Katie Hafner noted that students were "six deep" at the Google table at Stanford's recruiting fair. They remark, of Microsoft and Yahoo!: "Both companies are already fighting the perception that their most innovative days are behind them".

They go on to: "'Engineers here want to work on tomorrow's technology, not yesterday's", said Bill Demas, who worked at Microsoft through much of the 1990s and then at Yahoo until leaving last year. He is now chief executive at Moka5, a start-up of around 30 people based in Silicon Valley's Redwood City.'

"'If it's perceived that Yahoo or anyone else is not focused on the future, it's going to be very difficult to recruit top people,' Mr. Demas said."

Rivlin and Hafner continue: "One risk for Microsoft is that it could spend billions to buy Yahoo only to find that many of its most talented people have already left. That is one of the perils of high-priced acquisitions in the talent economy, where the real prize is often the collective abilities of a company's employees".

"Silicon Valley, after all, is a place where large companies have been known to pay vast sums for smaller ones largely to buy the commitment of a few gifted engineers. But like others interviewed for this article, Mr. Becker of New Cycle Capital said he was already hearing about Yahoo employees, loath to work for an even larger corporation whose bosses are around 850 miles away in Redmond, Wash., reaching out to friends working at other companies."

That's not hard. Between the time of the split between AT&T and Lucent and the actual closing of the Dept. 1127 (2005) Ken Thompson retired to California. Brian Kernighan is a Professor at Princeton. Doug McIlroy is a Professor at Dartmouth. Rob Pike and Dave Presotto and Sean Dorward are at Google. Tom Duff is at Pixar. Phil Winterbottom is CTO at Entrisphere. Gerard Holzmann is at NASA/JPL Lab for Reliable Software. Bob Flandrena is at Morgan Stanley.

A former employee at 1127 told me: "My take is that 1127 probably reached Schiavo status when Rob, Presotto, et al. fled west to Google". [Terri Schiavo was the 'brain dead' woman who became the centrepiece of a right-to-die battle.]

Well, the Yahoo! Board rejected the offer. And, as I write, we're all waiting for Microsoft's retaliatory strike.

But, instead of waiting, Bradley Horowitz, head of Yahoo's Advanced Technology Division, has already jumped ship for Google. How many more will there be?

But Microsoft hasn't been inactive. They've bought Danger, the maker of the T-mobile Sidekick, Paris Hilton's phone. And Yahoo!? On 13 February Yahoo! said it had acquired Maven Networks, an Internet company that sells a system for managing advertisements in online videos. And Google? On 17 February it was announced that Google was acquiring Pyra Labs, which "includes the Blogger 'push-button' Web publishing platform used by hundreds of thousands of users to update online journals, or blogs."

No grass is growing under their acquisitive feet.

The Utah trial of The SCO Group v Novell will be heard at 16:30 UMT on 29 April. If and when there's a conclusion, I will report..

By the time you read this, the world will know more about these hot topics.

Blatant self-advertising

As we achieve April First, get your favourite networker a copy of *The Complete April Fools' Day RFCs* ISBN-13: 978-1573980425. Only 15 pounds from amazon.co.uk!

If you're into history, get *The ARPANET Sourcebook*, ed. by P H Salus ISBN-13: 978-1573980005.

What Virtualization can do for Security

Tal Garfinkel and Andrew Warfield

Virtual Machine (VM) technology is rapidly gaining acceptance as a fundamental building block in enterprise data centers. It is most known for improving efficiency and ease of management. However, it also provides a compelling approach to enhancing system security, offering new ways to rearchitect today's systems and opening the door for a wide range of future security technologies.

Virtualization emerged as a technique for managing mainframes in the 1960s. In the late 1990s, it was rediscovered in the midst of a perfect storm. Widespread adoption of IT infrastructure based on inexpensive commodity PCs led to explosive growth in the number of machines in enterprise environments. Concurrently, intense growth in the commodity software industry left a legacy of large, feature-rich, and complex applications and operating systems – so complex, in fact, that the best practice for managing and securing them was (and still is) commonly held to be isolating each server application to its own, incredibly underutilized host.

Remedying this inefficiency through server consolidation is perhaps the most well known use of virtualization. More recently the benefits of virtualization for management, such as simplifying provisioning and allowing hardware upgrades without incurring downtime, are becoming equally well known. Another property of this technology that has received less attention is the benefits it can provide for security.

Our objective is to provide readers with a better sense of what virtualization can contribute in this area. We begin by looking at the basic security benefits VMs can provide today (e.g. its power as a mechanism for isolation). We then survey some of the emerging security technologies supported by virtualization that we may see in the years ahead.

Virtual Machines for Isolation

The oldest and simplest path to enhancing security with VMs is by separating multiple applications on a single OS into multiple VMs, with each application in its own VM. As with application sandboxes, ails, etc., this contains the damage an exploited application can inflict on other services. This works equally well in situations with stand-alone applications (e.g. protecting a DNS or email server rom a compromised Web server) and for services consisting of a more complicated aggregate of middleware, such as an e-commerce application with a Web server front end, back-end logic for dealing with user transactions, and a database – each of which could be run in its own VM.

This isolation presents two major benefits that we discuss in detail throughout this article. First, by virtue of running in separate virtual machines, applications are obviously much less vulnerable to compromises that start in other applications; that is, the result of a system compromise is better *confined* in a correctly configured virtualized environment. Second, many of the security mechanisms that we have today are best applied at a host granularity. The *encapsulation* afforded by more single-purpose VM-based environments allows much stricter security policies to be applied at a (virtual) host granularity.

Making machine-level isolation ubiquitous

Isolating applications on their own machine to contain a compromise (e.g., in a company's DMZ) has long been considered best practice. By reducing the physical and management costs of whole system isolation, this practice can be applied far more ubiquitously. Of course, replacing physical isolation with virtualization does come at some cost in assurance, and there are some situations (e.g., separating machines in the DMZ from those inside the firewall) where the additional security this affords makes sense. However, in the common case, the benefits of virtual-machine-monitor-based (VMM-based) isolation outweigh the resulting loss in assurance. A hybrid approach seems most sensible – for example, relying on a VMM to compartmentalize applications on either side of the firewall, while putting a physical gap between these two sides.

Running several virtual machines on a single platform, each hosting a single application, incurs only nominal overhead over running these applications all on the same OS. Depending on the applications and platform configuration, this overhead can potentially even be less, as on multicore platforms, and virtualization can sometimes make it easier to reduce resource contention and expose latent concurrency.

Similarly, application setup times on a virtualized platform are often less than on a traditional platform. Uniform virtual hardware means only a single "base" operating system image is required, which can then be specialize on a per-application basis. Increasingly, administrators will simply have totally configured application VMs on hand in "template" form, where a new VM (e.g., a mail server) can simply be instantiated and deployed as needed. Also gaining popularity is the use of a prebuilt virtual appliance (i.e., an application that has been bundled along with an operating system in a VM by the software vendor).

Why whole machine isolation?

Using virtual machines for isolation frequently elicits the question, "Why use something so coarse-grained?" BSD jails, application sandboxes such as Systrace or AppArmor, and fine-grained OS-level access controls such as SELinux have long been available. The simple answer is that VMs offer potential benefits over these solutions by way of simplicity and assurance, in terms both of implementation and of configuration.

Correctly utilizing OS-level isolation often requires a deep understanding of both OS semantics and the behavior of the particular application(s) being secured (e.g., file system usage), to securely configure a system and to debug reliability problems associated with configuration errors. In contrast, the basic abstraction that VMs provide is familiar and easy to understand – at most, an administrator must configure a firewall for containment or set up a network file system for sharing. Beyond these tasks, no OS-specific knowledge is required. Further, securing an OS against exploits is a demonstrably difficult problem, owing to their many functions and broad attack surfaces. The narrow interface that a virtual machine presents offers a comparatively harder target for attackers, thus potentially providing higher assurance isolation.

Of course, virtualization is not a substitute for OS-level mechanisms. If an application is compromised, using a defense-in-depth strategy to contain that compromise only improves protection. Thus, if an attacker must break out of a jail, sandbox, or other OS-based protection mechanisms before attempting to overcome the isolation imposed by the VM, all the better. Further, a VMM-based approach is not always suitable: VMMs excel at isolation, but in situations where there is a great deal of controlled sharing, an OS-level solution may be more suitable.

System Specialization

Beyond reducing the possibility that a compromise in one application can spread to another, putting each application in its own virtual machine conveys a variety of benefits. First, it eliminates complexity inside the guest, making access controls and other hardening measures easier to set up and allowing a thinner OS to be used, reducing the size of the application's trusted computing base. Next, each VM's attack surface can be reduced, as each VM only requires the interfaces (network ports, RPC daemons, etc.) a single application needs to be enabled.

This approach converges on a virtual appliance model. Just as on a hardware appliance, in a virtual appliance the operating system is specifically tailored from the ground up for the application that it's running. An example of the extreme end of this spectrum is BEA Systems Liquid VM [1], a custom operating system tailored specifically to run its JRocket JVM and middleware platform. Other vendors offer a middle ground, providing custom versions of existing operating systems specifically tailored to the needs of appliances. Both approaches offer the possibility of a smaller trusted computing base and reduced attack surface for applications. Additionally, this allows securing applications to shift from system administrators to ISVs, who can often use their greater knowledge of an application to employ more complex hardening measures.

Data Center Partitioning

Technologies for segmenting data centers, such as VLANS, have become increasingly compelling, as they provide an intuitive model of separation. Using virtual machines, this model of separation can be pushed onto the end host, making network-based partitioning an end-to-end proposition. For example, prior to its adoption of virtualization, the NSA used to use physically separate machines to access networks with data at different levels of classification. This provides excellent isolation, but of course an architecture like this is expensive and unwieldy. This prompted the NSA's move to their Nettop architecture [5], which instead used virtual machines for isolation on the same physical host.

With the cost of such an end-to-end solution reduced, such an architecture becomes feasible for normal businesses. One common pattern is to partition a user's desktop into two parts: One partition has access to all company internal resources, while the other can communicate with the outside world using Web browsers, IM, etc., with all the ensuing perils this entails. This pattern can also be applied inside a company, as with the NSA example. Organizational units such as marketing, engineering, and sales may be configured to have strongly isolated virtual resources, compartmentalizing these organizational roles in the face of compromise.

Virtualization facilitates another interesting sort of partitioning. As backup, logging and monitoring, remote display, and other functionality migrate out of the application VM and into separate protection domains on the virtualization layer, a natural separation occurs between the management plane (with all the infrastructure services that are important for security and management) and the application plane (with services actually used by end users). This again naturally supports an end-to-end separation of duties, with management functionality living on a separate set of virtual machines, separate network, etc., from running services – again, making it easier to gain confidence in the integrity of this part of a data center, even in the face of application compromise.

Virtual Machines for Fine-Grained Protection

Another unfortunate consequence of the rapid growth in size and complexity in commodity operating systems has been a predictable reduction in our ability to have faith that these systems cannot be compromised.

Virtualization can help us address this problem by adding additional protection for code running inside virtual machines. Virtualization may be used both to provide a higher-assurance protection layer than is afforded by the guest OS and to offer an additional degree of defense in depth, for example, preventing data from being leaked from a host, even if the guest OS is compromised.

Quis custodiet ipsos custodes?

The question, "Quis custodiet ipsos custodes?" ("Who will guard the guards?") is an important one for modern commodity operating systems. Today's antivirus and host-based intrusion detection and prevention systems are stuck with a difficult chicken-and-egg problem. Much of what they are seeking to detect is OS compromise, especially in light of the growing popularity of kernel rootkits. However, once the kernel has been compromised, the detection tools themselves are left unprotected. Thus, the arms race between attackers and defenders rapidly devolves into a complex game of core wars. This state of affairs has been institutionalized with the introduction of PatchGuard in Microsoft Windows Vista, and, unsurprisingly, the arms race to disable this mechanism is already in full swing [6].

Virtualization provides a way out of this situation by allowing the "guards" to be run in an entirely different protection domain, *outside* the influence of the OS that they are protecting. This can be accomplished in a number of ways. For example, a kernel rootkit detector could be protected by the VMM in situ (i.e., in the same address space as the operating system it is protecting) by preventing modifications to detector code and ensuring that it is executed with a specified frequency. Alternatively, the detector could be moved outside of the guest OS entirely and run in a totally separate VM [2]. Both approaches offer a simple and clear advantage over today's systems, where intrusion detection and prevention systems must paradoxically rely for protection on the OS they are trying to protect.

Getting Better HIPS

VMs can enhance monitoring and enforcement capabilities in AV and HIPS by allowing efficient interposition on hardware events in the guest OS. x86 virtualization has long played technically exciting tricks involving the virtualization of the hardware memory management unit (MMU); MMU virtualization is required to prevent VMs from accessing each other's memory, and providing high-performance implementations is one of the major challenges of effective x86 virtualization. Leveraging this capability for security, a system can enforce exactly what code should be permitted to execute on a host [4] or, alternately, perform up-to-the-moment detection of malware, offering superior detection capabilities when compared with the file-scanning approaches of today's AV systems. Interposing on devices offers many other possibilities: for example, preventing malware from being written to disk, scanning a USB disk when it is plugged into the machine, or filtering network traffic.

Locking Down Data

Because it acts as a small privileged kernel that runs under an existing guest OS, the virtual machine monitor can impose nearly arbitrary protection policies inside the guest. Potential applications range from protecting key data in an SSL stack, to preventing sensitive documents from being leaked from a VM, to enforcing fine-grained information flow policies. Although few technologies offering this capability have been deployed to date, the possibilities are rich and promising.

Logging and Snapshots: More of What You Want and Less of What You Don't

With the shift from physical to virtual machines, what was once hardware state, locked away on devices, becomes entirely software-based. This change allows us to rethink how we manage and take advantage of system state.

Down with Hard State

One of the biggest challenges in managing security within an installed base of software is keeping it secure as it ages. Users and administrators generally accept that a freshly installed application running on a freshly installed OS usually works just fine. However, as other software is installed and uninstalled, applications are run, and time passes, confidence in the safe and correct configuration of a system diminishes. Moreover, once a system has been compromised – or in many cases even *may have been compromised* – the only way to restore confidence is to return to a "clean" state, which is often achieved by completely reinstalling the OS, or at least reimaging the system.

VMs provide a very compelling way to deal with this sort of complexity. They can trivially checkpoint and return to points in history, allowing the unknown permutations to a system to be dropped. Further, we may specify a positive filter on what changes we *do* want to keep, perhaps preserving the contents of a bookmarks file while reverting a Web browser VM to a freshly installed state with each restart. In this sense, VMs allow us to treat millions of lines of complex software as a very coarse-grained transformation on a small and confined set of data. Any changes made by the VM to internal state other than the interested set can simply be dropped and reverted to a fresh install, providing much stronger levels of confidence in the system.

Forensics and Remediation in the log-structured Data Center

Carrying this notion of logging and reverting the state of a system one step further allows us to consider simply recording to a log everything any VM in a data center ever does. Considerable research work on virtualization has explored the notion of logging and replaying a VM's execution at a fine granularity, using techniques as fine-grained as cycle-accurate instruction replay or as loose as logging network traffic and periodic VM checkpoints. Regardless of the technique used, it's quite reasonable to imagine production systems that include a facility to return to arbitrary points in the history of their execution.

This detailed logging has the potential to solve the very challenging task of separating good from bad in an exploited system. In normal situations, once a system is found to have been compromised the best that an administrator can possibly do is to revert to a backup and attempt to comb through the file system, searching for changes and determining which should be preserved. More often, this represents an unrealistic effort and the post-backup changes are simply lost.

Having a detailed log of a system's execution allows a compromise to be analyzed retrospectively. Detailed analysis tools may be built and run against the system's execution log and attempts can be made to isolate malicious changes, improving our ability to recover data. As a gedanken experiment for what this means in restoring compromised systems, imagine the ability to rewind the execution of a system to just before the point that it was attacked and there insert a firewall rule that refuses to admit the exploit traffic. The system would then play forward without maliciousness, ideally preserving a considerable amount of "good" activity.

Logging and analysis similarly provides the ability to more speedily understand malicious software and attacks on systems, because it allows forensic analysis to be run both forward and backward in time to diagnose the root of a system compromise and determine what malicious activities took place.

The Managed Desktop

The desktop is an inevitable final frontier for virtualization. Its support for a wide range of hardware and the latency-sensitive nature of its applications have posed a higher technical barrier for desktop entry than for the server space. However, from a security perspective the rewards are high: Desktop systems are exactly where many of the advantages that we have described here are most desirable.

In addition to richer policies, virtualization of the desktop allows security policies to be applied uniformly across the enterprise. Firewall policies, network quarantine, monitoring, and the like can be applied whether the user is at his or her desk or in the data center, regardless of the integrity of the guest OS.

Virtualization also provides the ability to strictly limit the actual hardware to which applications contained in VMs have access. For example, the discreet use of a USB memory stick to transfer applications or data off of a machine may be disallowed, a VM's access to the network may be very tightly controlled, and VM data stored on disk can be automatically encrypted.

A Brave New World

We believe the benefits of virtualization for efficiency, platform flexibility, and ease of management alone will make it ubiquitous in enterprise data centers. As with provisioning and management before it, the benefits of virtualized platforms for improving security will take time to be realized. Part of the current unrealized potential is a lack of deep operational experience in modern IT environments, widespread understanding of how this technology can be leveraged, and tools that help facilitate best practices.

Our enthusiastic endorsement of virtualization's potential should be tempered with the observation that the flexibility and power that make it such a boon for system management also give rise to a variety of new security challenges [3]. Coping with these will require a varied combination of new technologies and best practices. As always, the responsibility for ensuring the secure adoption of virtualized platforms will lay in the hands of both platform vendors and those deploying them.

REFERENCES

[1] BEA Systems, Inc., "Stop Worrying About Computing Capacity – and Start Making the Most of It," 2007:
`http://www.bea.com/content/news_events/white_papers/BEA_Virtualization_wp.pdf`

[2] T. Garfinkel and M. Rosenblum, "A Virtual Machine Introspection Based Architecture for Intrusion Detection," Proceedings of the Network and Distributed Systems Security Symposium, February 2003.

[3] T. Garfinkel and M. Rosenblum, "When Virtual Is Harder Than Real: Security Challenges in Virtual Machine Based Computing Environments," Proceedings of the 10th Workshop on Hot Topics in Operating Systems (HotOS X), May 2005.

[4] L. Litty and D. Lie, "Manitou: A Layer-Below Approach to Fighting Malware," Proceedings of the Workshop on Architectural and System Support for Improving Software Dependability (ASID), October 2006.

[5] R. Meushaw and D. Simard, "NetTop: Commercial Technology in High Assurance Applications," 2000:
`http://www.vmware.com/pdf/TechTrendNotes.pdf`

[6] Uninformed, "PatchGuard Reloaded. A Brief Analysis of PatchGuard Version 3," September 2007:
`http://uninformed.org/index.cgi?v=8&a=5`

Originally published in ***;login: The USENIX Magazine****, vol. 32, no. 6 (Berkeley, CA: USENIX Association, 2007).*

IDS signature matching with iptables, psad, and fwsnort

Mike Rash

The analysis of log data is becoming an increasingly important capability as more applications generate copious amounts of run-time information. This information often has interesting things to say for those who are listening (including evidence of events that are significant from a security perspective), but the sheer volume of information often requires automated tools to make sense of the data. The iptables firewall is built on top of the Netfilter framework in the Linux kernel, and it includes the ability to create verbose syslog messages of the network and transport layer headers associated with IP packets. In addition, through the use of the iptables string match extension, the application layer can be searched for evidence of malicious activity and iptables can then log or take action against such packets.

This article explores the use of psad and fwsnort [1] to automate the analysis of iptables log messages with a particular emphasis on passive OS fingerprinting and the detection of application-layer attacks. Both psad and fwsnort are open-source software released under the GNU Public License (GPL). Some familiarity with iptables and the Snort rules language is assumed in this article [2]. Also, see the INSTALL file bundled with the psad and fwsnort sources for installation instructions.

Network Setup and Default iptables Policy

I will illustrate network traffic against a Linux system that is protecting a small internal network with an iptables policy that implements a default “log and drop” stance for any traffic that is not necessary for basic connectivity. In particular, the iptables policy provides NAT services to allow clients on the internal network to issue DNS and Web requests out through the firewall (with the internal network having the RFC 1918 subnet 192.168.10.0/24 and the external interface on the firewall having a routable IP address), and the firewall accepts SSH connections from the internal network. The iptables policy uses the Netfilter connection tracking capability to allow traffic associated with an established TCP connection to pass through; also allowed are packets that are responses to UDP datagrams (which may include ICMP port unreachable messages in response to a UDP datagram to a port where no server is bound). All other traffic is logged and dropped (with iptables log messages reported via the kernel logging daemon klogd to syslog). This iptables policy is implemented by the following iptables commands [3]:

```
iptables -F INPUT
iptables -P INPUT DROP
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -i eth1 -p tcp -s 192.168.10.0/24 - -dport 22 -m state \
--state NEW -j ACCEPT
iptables -A INPUT -i !lo -j LOG --log-ip-options --log-tcp-options - -log-prefix "DROP"
iptables -F FORWARD
iptables -P FORWARD DROP
iptables -A FORWARD -m state - -state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -p tcp -s 192.168.10.0/24 - -dport 80  -m state --state NEW -j ACCEPT
iptables -A FORWARD -p tcp -s 192.168.10.0/24 - -dport 443 -m state --state NEW -j ACCEPT
iptables -A FORWARD -p udp -s 192.168.10.0/24 - -dport 53 -j ACCEPT
iptables -A FORWARD -i !lo -j LOG --log-ip-options --log-tcp-options --log-prefix "DROP"
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.10.0/24 -j MASQUERADE
```

Passive OS Fingerprinting

With the iptables policy active on the Linux system, it is time to see what it can show us from a logs perspective. First, from a system on the internal network (with hostname `int` and IP address 192.168.10.50), we attempt to initiate a TCP connection to port 5001 on the firewall (where the firewall’s internal IP is 192.168.10.1):

```
[int]$ nc 192.168.10.1 5001
```

This results in the following iptables log message for the incoming TCP SYN packet, which is blocked and logged by iptables:

```
Sep 13 21:22:24 fw kernel: DROP IN=eth1 OUT=                                 \
MAC=00:13:46:3a:41:4b:00:0c:41:24:56:37:08:00                                \
SRC=192.168.10.50 DST=192.168.10.1 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=51104 DF \
PROTO=TCP SPT=57621 DPT=5001 WINDOW=5840 RES=0x00 SYN URGP=0                 \
OPT (020405B40402080A1ECB4C4C0000000001030302)
```

The log message contains, among other things, source and destination IP addresses, the IP `ID` and `TTL` values, source and destination port numbers, TCP flags (with just the `SYN` flag being set in this case), and the options portion of the TCP header (preceded by the `OPT` string). From the perspective of passively fingerprinting the operating system that generated the TCP `SYN` packet against the firewall, the most interesting fields in the log message are as follows:

```
IP length: LEN=60
TTL: TTL=64
The Don't Fragment bit: DF
TCP window size: WINDOW=5840
TCP flags: SYN
TCP options: OPT (020405B40402080A003F83040000000001030302)
```

(Note that the TCP options string is only included within an iptables log message if `--log-tcp-options` is given on the iptables command line when adding the `LOG` rule.) These fields are important because they are the same fields that the best-known passive OS fingerprinting software, p0f, uses to fingerprint operating systems [4]. This illustrates the completeness of the iptables logging format, because it is possible to implement the same passive OS fingerprinting algorithm used by p0f but use iptables log messages as input instead of sniffing packet data off the wire with a packet capture library. The psad project implements the p0f fingerprinting algorithm over iptables log messages, and the TCP log message just listed conforms to the following p0f fingerprint:

`S4:64:1:60:M*,S,T,N,W2:Linux:2.5::Linux 2.5` (sometimes 2.4)

This fingerprint specifies a series of requirements on packet headers separated by colons and is read as follows: `S4` requires that the TCP window size be four times as large as the Maximum Segment Size (MSS). The MSS value is part of the TCP options field. `64` matches the `TTL` value and requires that the initial `TTL` is 64. (This value has to be estimated for packets that traverse the open Internet.) `1` requires that the Don't Fragment bit is set. `60` requires that the overall size of the `SYN` packet (including the IP header) be 60 bytes. `M*,S,T,N,W2` describes the options field of the TCP header; `M*` means any MSS size, `S` means Selective Acknowledgment is OK, `T` means that the TCP options contain a time stamp, `N` requires a NOP option, and `W2` requires a window scaling value of 2.

Decoding the options string from the iptables log message is the most complex portion of the fingerprinting activity. The options string follows Type Length Value (TLV) encoding, where each TCP option has one byte for the option type, one byte for the option length, and a variable number of bytes for the option value [5]. Hence, the options string decodes to the following, which matches the requirements of the `Linux:2.5::Linux 2.5` p0f signature (and psad reports this fingerprint within email alerts that it generates [6]):

```
MSS: 1460
Selective Acknowledgment is OK
Timestamp: 516639820
NOP
Window scaling value: 2
```

Snort Rule Matching with fwsnort

In the previous section, we saw that it is possible to collect iptables log messages for SYN packets sent from arbitrary hosts and, in many cases, infer the OS that generated these packets. Passively fingerprinting operating systems is a nice trick and can reveal interesting information about an attacker, but in the threat environment on the Internet today the real action is at the application layer (OS fingerprinting only requires the inspection of network and transport layer headers). To get a feel for how important application-layer inspection is to computer security, one need only examine the Snort rule set. In Snort version 2.3.3 (the last version of Snort that included rules released under the GPL instead of the VRT service from Sourcefire), there are about 150 signatures out of 3,000 that only test packet headers and have no application-layer match requirement.

In the Snort rules language, elements that test the application layer include the `content`, `uricontent`, `pcre`, `byte_test`, `byte_jump` and `asn1` keywords, whereas elements such as `flags`, `ack`, `seq` and `ipopts` (among others) test packet header fields. Maintaining an effective intrusion detection stance for network traffic requires the ability to inspect application-layer data, and 95% of all Snort rules are focused on the application layer.

The fwsnort project translates Snort rules into iptables rules that are designed to detect (and optionally react to) the same attacks, and the Snort 2.3.3 rule set is packaged with fwsnort. Because the detection capabilities of iptables are limited to matches on strings via the string match extension [7], many Snort rules (such as those that contain a pcre match) cannot be translated. Still, about 60% of all Snort 2.3.3 rules can be translated into iptables rules by fwsnort because iptables provides a flexible set of facilities to the user for matching traffic in kernel space. Chief among these facilities is the ability to match on multiple content strings instead of just a single string; iptables 1.3.6 introduced this capability by allowing multiple matches of the same type to be specified on the iptables command line. In the following we will see an example of a Snort rule that looks for two malicious content strings returned from a Web server and will see how iptables can be made to look for the same attack in network traffic.

Some of the most interesting and devastating attacks today exploit vulnerabilities in client applications that are attacked via malicious or compromised servers. Because in many cases thousands of independent client applications communicate with popular servers, an attacker can take advantage of this multiplying effect just by compromising a heavily utilized server and forcing it to launch attacks against any hapless client who connects to it.

An example of a Snort rule that looks for a client-side attack against a Web browser is rule ID 1735, which is labeled as `WEB-CLIENT XMLHttpRequest attempt`. This rule detects a possible attempt to force a Web browser to return a list of files and directories on the system running the browser back to the attacker, via the responseText property, after redirecting the browser to point to the local filesystem. Fortunately, this attack applies to older versions of the Netscape and Mozilla browsers, but if a Web server sends data that matches this Snort rule back to a Web browser running on my network, I would want to know about it regardless of whether or not the browser is vulnerable. The XMLHttpRequest attack is tracked in the Common Vulnerabilities and Exposures (CVE) database as CVE-2002-0354 [8]. Here is the Snort rule for this attack:

```
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any            \
(msg:''WEB-CLIENT XMLHttpRequest attempt'';                     \
flow:to_client,established; content:''new XMLHttpRequest|28|''; \
content:''file|3A|//''; nocase; reference:bugtraq,4628;         \
reference:cve,2002-0354; classtype:web-application-attack;      \
sid:1735; rev:7;)
```

Note that the Snort rule is looking for two content strings that emanate from an external Web server (with the source IP being `$EXTERNAL_NET` and the source port being `$HTTP_PORTS`) back to a Web client that is on the internal network (with the destination IP being `$HOME_NET` and the destination port being `any`, since the local TCP stack would choose a random high port for the Web session). The two content strings are `"new XMLHttpRequest|28|"` and `"file|3A|//"`. Each of these strings specifies one byte by its hex code between pipe characters: `|28|` in the first content string, and `|3A|` in the second. So, when translating this Snort rule into an iptables rule, we must account for that. With fwsnort installed, let's use it to translate Snort rule ID 1735 and then load it into the iptables policy on the firewall (some output below has been abbreviated):

```
[fw]# fwsnort --snort-sid 1735
[+] Parsing Snort rules files...
[+] Found sid: 1735 in web-client.rules
[+] iptables script: /etc/fwsnort/fwsnort.sh
[fw]# /etc/fwsnort/fwsnort.sh
[+] Adding web-client rules.
```

Examine the `/etc/fwsnort/fwsnort.sh` script and you can see the iptables command below. This command uses the `--hex-string` argument so that the Snort content fields can be specified "as is" within the iptables command (with the bytes between the pipe characters being properly interpreted), and the rule target instructs iptables to log any matching packet with the prefix `[1] SID1735 ESTAB`. This prefix informs the user that Snort rule ID 1735 was detected within an established TCP connection (fwsnort interfaces with the Netfilter connection tracking capability for this), and the rule is the first rule `[1]` within the `FWSNORT_FORWARD_ESTAB` chain.

The `--algo bm` argument instructs the string match extension to use the Boyer-Moore string-matching algorithm to conduct the application-layer match. With kernels in the Linux 2.6 series, the string match extension leverages a text-matching infrastructure implemented in the kernel which supports multiple string-matching algorithms; the Boyer-Moore algorithm exhibits excellent performance characteristics and is commonly used within open source and proprietary intrusion detection systems. Finally, the iptables comment match is used to include the Snort rule `msg`, `classtype`, and `reference` fields within the iptables rule for easy viewing under a command such as `iptables -v -n -L FWSNORT_FORWARD_ESTAB`.

We then have:

```
$IPTABLES -A FWSNORT_FORWARD_ESTAB -d 192.168.10.0/24 -p tcp          \
--sport 80 -m string --hex-string "new XMLHttpRequest|28|"            \
--algo bm -m string --hex-string "file|3A|//" --algo bm -m comment    \
--comment "sid:1735; msg:WEB-CLIENT XMLHttpRequest attempt;           \
classtype:web-application-attack; reference:bugtraq,4628; rev:7;      \
FWS:1.0.1;"                                                           \
-j LOG --log-ip-options --log-tcp-options --log-prefix                \
"[1] SID1735 ESTAB "
```

Now let us simulate the XMLHttpRequest attack through the iptables firewall against an internal Web browser. For this, we use Perl and Netcat on a dummy Web server at IP 11.11.1.1 (a randomly selected IP address for illustration purposes only). The following Perl command sends data matching the two content fields in Snort rule ID 1735 back to the Web client as soon as it connects:

```
[webserver]# perl -e 'printf "new XMLHttpRequest\x28AAAAAAAfile\x3A//"' |nc -l -p 80
[int]$ nc -v 11.11.1.1 80
Connection to 11.11.1.1 80 port [tcp/www] succeeded!
new XMLHttpRequest(AAAAAAAfile://
```

The last line here shows that the Web client received data that matches the Snort rule; iptables has not interfered with the traffic and has happily let it pass into the internal network. On the firewall, we see the following iptables log message (note that the `[1] SID1735 ESTAB` log prefix and the `ACK` and `PSH` flags are set, since this packet was matched within an established TCP connection):

```
Sep 14 08:39:24 fw kernel: [1] SID1735 ESTAB IN=eth0 OUT=eth1     \
SRC=11.11.1.1 DST=192.168.10.50 LEN=85 TOS=0x00 PREC=0x00 TTL=63  \
ID=23507 DF PROTO=TCP SPT=80 DPT=34646 WINDOW=91 RES=0x00 ACK PSH \
URGP=0 OPT (0101080A650A550A1F663D7A)
```

At this point we are confident that iptables is able to detect the attack. However, because iptables is a firewall, it is also inline to the traffic whereas Snort (unless deployed in inline mode) is merely able to passively monitor the traffic. Let us take advantage of this by changing the fwsnort command. This time we use the `--ipt-reject` command-line argument to have fwsnort use the iptables `REJECT` target against the Web connection in order to knock it down with a TCP `RST` packet:

```
[fw]# fwsnort --snort-sid 1735 --ipt-reject
[+] Parsing Snort rules files...
[+] Found sid: 1735 in web-client.rules
[+] iptables script: /etc/fwsnort/fwsnort.sh
[fw]# /etc/fwsnort/fwsnort.sh
[+] Adding web-client rules.
```

Let us run the attack simulation once more:

```
[webserver]# perl -e 'printf "new XMLHttpRequest\x28AAAAAAAfile\x3A//"' |nc -l -p 80
[int]$ nc -v 11.11.1.1 80
Connection to 11.11.1.1 80 port [tcp/www] succeeded!
```

We see that the client is again able to successfully establish a TCP connection with the Web server (that is, the TCP three-way handshake is allowed to complete), but no data comes across. This is because of the TCP `RST` generated by the `REJECT` target against the Web server. The `REJECT` target only sends the `RST` to the IP address that triggered the rule match within iptables, so the Web client never sees it. However, the iptables `REJECT` target is a terminating target, so it also drops the matching packet (in this case the packet that contains the XMLHttpRequest string). Hence, the malicious traffic never makes it to the targeted TCP stack, and this is an important capability when; some attacks only require a single packet in order to do their dirty work (the SQL Slammer worm is a good example). Only an inline device can prevent individual malicious packets from reaching their intended target.

On the firewall, fwsnort has also created a logging rule that produces the following log message (note that the log prefix now includes the string `REJ`, indicating that the packet was rejected):

```
Sep 14 08:41:24 fw kernel: [1] REJ SID1735 ESTAB IN=eth0 OUT=eth1  \
SRC=11.11.1.1 DST=192.168.10.50 LEN=85 TOS=0x00 PREC=0x00 TTL=63   \
ID=46352 DF PROTO=TCP SPT=80 DPT=52078 WINDOW=91 RES=0x00 ACK PSH \
URGP=0 OPT (0101080A650ACA031F66B26C)
```

Conclusion

This article has focused on two relatively advanced usages of functionality provided by the iptables firewall: the completeness of the log format, which makes passive OS fingerprinting possible, and the ability to inspect application-layer data for evidence of malicious activity. The psad and fwsnort projects automate both of these tasks and can provide an important additional security layer to an iptables firewall. The Snort community has guided the way to effective attack detection on the Internet today, and iptables can leverage the power of this community to extend a filtering policy into the realm of application inspection. Armed with such a policy, iptables becomes a sentry against application-layer attacks.

REFERENCES

[1] See:
`http://www.cipherdyne.org/`

[2] Snort rule writing documentation:
`http://www.snort.org/docs/writing_rules/chap2.html`

[3] A script that implements the default iptables policy can be downloaded from
`http://www.cipherdyne.org/LinuxFirewalls/ch01/iptables.sh.tar.gz`

[4] Passive OS fingerprinting is really passive stack fingerprinting. That is, the IP and TCP stacks used by various operating systems exhibit slight differences, and detecting these differences (i.e., "fingerprinting" the stack) can allow the operating system that uses the stack to be guessed; see
`http://lcamtuf.coredump.cx/p0f.shtml`

[5] There are two exceptions to this: The "NOP" and "End of Option List" options are only one byte long. See RFC 793 for more information.

[6] For examples of such alerts, see:
`http://www.cipherdyne.org/psad/docs`

[7] The iptables u32 extension is being reintegrated with the Netfilter framework after it was deprecated late in the 2.4 series kernel, so more complicated arithmetic tests can be written against both packet headers and application-layer data. The u32 extension essentially emulates the "byte_test" operator in the Snort rules language.

[8] See
`http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-0354`

Originally published in ***;login: The USENIX Magazine****, vol. 32, no. 6 (Berkeley, CA: USENIX Association, 2007).*

X Power Tools

Chris Tyler
O'Reilly Media
ISBN: 978-0-596-10195-4
270pp.
£ 24.99
Published: 11th January 2008

reviewed by Roger Whittaker

I found this book very interesting while at the same time wondering exactly what the intended market for it is. One section in the book is entitled "The Unused Toolbox" and describes some of the command line tools that come with the standard xorg or XFree86 distributions. In a way that title would make quite a good slogan for the book as a whole: it provides a source of detailed information about X, but at the same time much of that information is rather obscure and specialised.

For instance, I vaguely knew that there was a key combination that enabled "mouse keys", but I had no idea what it was (it's `Shift+Numlock`). Similarly I knew that you could run multiple X servers at the same time, but I didn't know the magic to make kdm or another display manager always start more than one X server at startup time: I do now (and I like this one as I happen to have a good use for it).

There's some good information about detailed X.org configuration, but I suspect (based on prior experience rather than testing the detailed information here) that some of the suggestions here will not always play well with automatically generated configuration files from particular Linux distributions.

The book contains some very good historical information about the development of X, and contrasts what the author calls "Old X" with the "New X" that has come about as a result of the explosion of new work in the Linux world since around the turn of the century.

There is useful information about VNC, including a "how to" on embedding an X application in a web page using VNC's java capabilities. There is also some good information about setting up a "kiosk" environment, but probably not enough to really get you to where you want to be if you really need to do this.

The omission of any discussion of FreeNX is a perhaps a pity, although it could be regarded as technically beyond the book's scope.

In general this is a technically well written book, which collects together a lot of good and useful information in one place (some of which is hard to find elsewhere), but perhaps does not quite reach the level required for the specialist readers who really need to buy a book about X.

Using Moodle

Jason Cole and Helen Foster
O'Reilly Community Press
ISBN: 978-0-596-52918-5
282pp.
£ 24.99
Published: 7th December 2007

reviewed by David Cohen

Moodle, as the book subtitle tells us, is the popular course management system. As a jobbing university teacher I have been using Moodle for content delivery for several years now and thought it was going quite well. This book made it clear to me how much I was missing.

The first three chapters are: Introduction, Moodle Basics and Creating and Managing Content. These get you started and whet your appetite for more detail.

Chapter four is called Managing your Class and tells you how Moodle roles can be creatively used to allow more interesting teaching and learning. Here you find out about Moodle's roles and capabilities, which are powerful enough to allow you to be really creative. I discovered that I could override basic permissions for students to allow them to grade each others posts on a particular Moodle forum inside my course. This was a revelation and the very next day I set up such a forum. Chapter four also discusses Moodle groups, which allow you to divide up your class. You can restrict how different groups interact within each activity on your Moodle course. For instance, if you enable groups, you get to create a per-group Wiki that students can use to help with online collaborative work. Also a per-group forum gives project team members a private place for thoughtful discussion and can even be used a simple online file repository.

The bulk of the book is concerned with describing the activities and resources with which you can populate your Moodle site. There are lots of chapters because Moodle is a very rich environment: Wikis, blogs, journals, quizzes, lessons, databases, assignment, glossaries etc. What was particularly nice about these chapters is that each finishes with a section giving some effective practises to help you deploy the Moodle features. An example I used straightaway was to create a glossary students can edit (with moderation). Moodle automatically links glossary terms from anywhere else in your course if you ask it do – great. Moodle is extensive so any book will miss out some stuff that you end up using. I use a course diary block on my courses and this is not described in the book. It is, however, very easy. Perhaps a more important deficiency is that there is very sparse material on Moodle setup and administration. Moodle is likely to sit in amongst a quite complicated set of tools for managing students, libraries, internal web portals, etc. However, in its defence, this book is aimed at teachers, not at administrators.

I particularly liked the style: information you need given succinctly; the text is clear and chatty. I particularly enjoyed the asides describing the two authors' experience of Moodle, including pitfalls to avoid and tricks they have deployed. Necessarily, this book is a bit repetitive as each new feature merits the same kind of discussion. That said, I read it from cover to cover in three days. If you are web savvy you probably won't need the book because the Moodle interface really is easy to use and all the information is online anyway. If, like me, you are the sort of person who likes a book to carry with them on trains, bending the pages and making notes, then this is a very good book on Moodle. It gives clear instructions for using features, good illustrations, useful tips. Maybe, once you've set up a course or two, you'll know exactly what to do and won't need the book, but until then…

I wish that I had read this book before I started using Moodle.

Apache Cookbook

Ken Coar and Rich Bowen

O'Reilly Media

ISBN: 978-0-596-52994-9

306pp.

£ 21.99

Published: 11th January 2008

reviewed by Paul Waring

As someone responsible for Apache installations on Windows, Linux and OS X, a book which claims to include hundreds of solutions to problems which I'm likely to face with this particular piece of server software certainly piques my interest.

Although the table of contents lists a wide array of topics, the most useful chapter in this book, at least in my opinion, is the one concerning aliases and URL rewriting. This is by far and away the most common problem which I experience with Apache, and judging by the number of questions related to it on forums and mailing lists I'm not alone. Fortunately, this chapter covers a lot of the common questions which people come across when trying to get to grips with mod_alias and mod_rewrite. My only minor complaint with this chapter is its size – twenty pages for one of the most difficult topics is insufficient. On the plus side, the chapter on security is fairly comprehensive, but then the separate Apache Security book has already covered that base.

My main concern about Apache Cookbook, however, is its usefulness to the average Apache administrator. Most people will be running a fairly standard Apache installation, perhaps with a few modules enabled, and so the vast majority of the recipes presented within the book will never be consulted. I myself run Apache on the three major operating systems and for different purposes, but the configuration is very similar on each one. Unless you are responsible for a web farm running Apache in a complex environment or have particularly exotic requirements, this book is unlikely to solve many of your problems.

The lack of a recipe for installing and configuring Ruby on Rails to work under Apache also felt like a bit of a let down to me. Admittedly the authors cannot cover every scenario, but the sheer number of Ruby on Rails books out there (many published by O'Reilly), coupled with the fact that there are not many UK web hosts offering this functionality, meant that the complete omission of Ruby was both a surprise and a disappointment.

Finally, most of the recipes found in this book can also be obtained for free following a quick search on Google or by consulting the Apache documentation files. Having all the answers in one book can be useful, but I'm not convinced that this makes up for the fact that the pace of software development means that some recipes may well be out of date by the time the book leaves the printers.

Overall, I wasn't impressed by this book. Too many of the recipes cover trivial problems (e.g. how to log the browser's user agent or installing Apache on Windows), others are covered in the documentation already supplied with Apache. Were this a first edition, I could almost forgive some of these mistakes, but not in a book which has already been through that phase. There's some useful material in here, but not enough to justify the cover price – especially when you can get a lot of the information from a quick search on Google. As such, it only scores 4/10 from me.

Linux Networking Cookbook

Carla Schroder
O'Reilly Media
ISBN: 978-0-596-10248-7
638pp.
£ 27.99
Published: 14th December 2007

reviewed by Paul Waring

Ninety percent of the time, the systems I run have no networking problems, receive most of their configuration by DHCP and Just Work. However, things can and do go wrong now and then, and it's at those times when I desperately need to know why my routing tables are broken or my ethernet interface won't come up that I look around for a concise explanation of how to fix the problem. Linux Networking Cookbook claims to cover "everything you need to know to excel as a Linux network administrator", which sounds just what I'm looking for.

The book is split into logical chapters, each of which covers a specific topic such as remote administration using ssh or setting up a Linux wireless router. In addition, the chapters themselves are also carefully constructed so that each solution follows on from the next. For example, the VoIP chapter begins with installing Asterisk from source or binaries, and the future solutions build upon this in a step by step way, finishing with more complex tasks such as connecting users on the move to the server. As a result of this structure, the reader can choose to either read the chapter from start to finish, picking up useful tips on the way, or jump straight to the solution for their particular problem if they've already got a working setup.

The chapter which I personally found to be most useful covers how to build virtual private networks with OpenVPN. Although one or two of the recipes are somewhat tenuous (e.g. configuring OpenVPN to start at boot, which is a fairly trivial task and also highly distribution-dependent), there is enough information to get you from a basic installation to having clients connect and authenticate using keys and certificates. Alas, I had already spent a significant amount of time wrestling with Google and the openvpn.net documentation by the time I got round to reading this book, but I cannot fault the authors for this situation!

One criticism which I will sometimes level at cookbooks is that most of the solutions within them can be found with a quick search on Google, and in many ways this is true for Linux Networking Cookbook. However, when your network card is down – particularly if it's the one in your router and/or firewall – and you need to find the right command to diagnose and fix the problem, I can see this book being a godsend.

The only minor niggle which I have with this book is that I was able to spot a few mistakes or omissions as I flicked through the recipes. Some of these were in places where I felt that a recipe had been hastily inserted in order to make up a chapter, without being properly worked out. A few other recipes seemed somewhat redundant, for example over half a page is devoted to installing a Windows program which boils down to downloading a setup file and installing it just like any other application. However, despite these defects, this is the kind of book which I expect to remain on my desk rather than my bookcase. With a bit of judicious editing and the removal of some redundant recipes, this could be a must-buy for every Linux sysadmin. Until then, it still scores a worthy 8/10.

Visualizing Data

Ben Fry

O'Reilly Media
ISBN: 978-0-596-51455-6
382pp.
£ 24.99
Published: 4th January 2008

reviewed by Lindsay Marshall

There are many books on visualising data that show wonderful examples of great visualisations, but none of them provide any useful help for people who have a lump of data that they need to display effectively. Creating great visualisations is a hard job that needs a lot of design skill and a huge chunk of luck to help you find the right way to go. I had hopes that this book would be more helpful but sadly it was not to be. It turns out that this is a manual for a programming environment for creating visualisations called Processing. Now, I have to confess to having never heard of Processing so I searched and found the web page and that

> *"Processing is an open source programming language and environment for people who want to program images, animation, and interactions. It is used by students, artists, designers, researchers, and hobbyists for learning, prototyping, and production. It is created to teach fundamentals of computer programming within a visual context and to serve as a software sketchbook and professional production tool. Processing is developed by artists and designers as an alternative to proprietary software tools in the same domain."*

Sounds great and it produces great output, but this book isn't really about creating great output (though it claims to be). It's about the nitty gritty, details of the programming language used to generate the output. It's about the brushstrokes and paints you need to paint the picture not about deciding what to paint or the composition. And the language itself? C. The book says that it is based on Java, but anyone who knows code will look at it and see C. Indeed they will see a sea of C. Page after page of raw code examples.

I'm pretty sure that if you have a need to use Processing then this will be a useful book (though I think it is a little too code heavy for my taste), but it's not what I was looking for, nor will be what many people guessed it to be from the title which really ought to be a bit more up front about the actual content.

Head First JavaScript

Michael Morrison

O'Reilly Media
ISBN: 978-0-596-52774-7
650pp.
£ 24.99
Published: 11th January 2008

reviewed by Lindsay Marshall

I think I must have slipped over the edge into old-fogeydom (though see below).

This book "uses a visually rich format designed for the way your brain works". Well, they must have different brains to me. I hate it. The "visually rich format" (or mess on a page) is noisy and full of irrelevance. It smacks of trendy vicars and guitars. I passed this book around in a tutorial session with my students and asked for their opinion – without exception they said it was awful. They hated the pictures, they hated the layout and they all said they would rather have a thinner (cheaper!) book that was text based.

If you scrape of the layers of cruft, there is nothing at all wrong with the the material being taught – all sensible and useful stuff – it's just the medium, not the message. I know something about learning theory and I appreciate what they are trying to do with Head First series, but it is just horrible, horrible, horrible. And the thing that I hated most of all was that sections labelled "Bullet points" had a picture of a bullet at the top. Lord. Privy. Seal.

Contributors

David Cohen is a professor of Computer Science at Royal Holloway. He has taught undergraduates for 20 years and is an active developer of automated submission and assessment software for programming assignments. He has been using Moodle for three years.

Tal Garfinkel is part of the advanced development group at VMware and is currently on leave from Stanford University, where he plans to submit his Ph.D. thesis any day now.

Lindsay Marshall developed the Newcastle Connection distributed UNIX software and created the first Internet cemetery. He is a Senior Lecturer in the School of Computing Science at the University of Newcastle upon Tyne. He also runs the RISKS digest website and the Bifurcated Rivets weblog.

Jane Morrison is Company Secretary and Administrator for UKUUG, and manages the UKUUG office at the Manor House in Buntingford. She has been involved with UKUUG administration since 1987. In addition to UKUUG, Jane is Company Secretary for a trade association (Fibreoptic Industry Association) that she also runs from the Manor House office.

Michael Rash holds a Master's degree in Applied Mathematics and works as a Security Architect for Enterasys Networks, Inc. He is the creator of the cipherdyne.org suite of open source security tools and is author of the book *Linux Firewalls: Attack Detection and Response with iptables, psad, and fwsnort*, published by No Starch Press.

Peter H Salus has been (inter alia) the Executive Director of the USENIX Association and Vice President of the Free Software Foundation. He is the author of *A Quarter Century of Unix* (1994) and other books.

Andrew Warfield has completed his PhD at the University of Cambridge and now divides his time between XenSource and an Adjunct Professor position at the University of British Columbia. He lives in Vancouver.

Paul Waring is a PhD student at the University of Manchester, researching ways into which events can be detected and linked on the Web. He occasionally emerges from the latest ACM conference proceedings to campaign on environmental issues, work on several writing projects and continue his freelance IT work. Paul is also the newest member of the UKUUG council, and looking forward to getting more involved in the organisation's work on a strategic level.

Alain Williams is UKUUG Chairman, and proprietor of Parliament Hill Computers Ltd, an independent consultancy.

Roger Whittaker works for Novell Technical Services at Bracknell and is the UKUUG Newsletter Editor.

Contacts

Alain Williams
Council Chairman
Watford
Tel: 07876 680256

Sam Smith
UKUUG Treasurer; Website
Manchester

John M Collins
Council member
Welwyn Garden City

Phil Hands
Council member
London

John Pinner
Council member
Sutton Coldfield

Howard Thomson
Council member
Ashford, Middlesex

Paul Waring
Council member
Manchester

Jane Morrison
UKUUG Secretariat
PO Box 37
Buntingford
Herts
SG9 9UQ
Tel: 01763 273475
Fax: 01763 273255
`office@ukuug.org`

Sunil Das
UKUUG Liaison Officer
Suffolk

Zeth Green
UKUUG Events Co-ordinator
Birmingham

Roger Whittaker
Newsletter Editor
London