SUDO MASTERY

ACCESS CONTROL FOR REAL PEOPLE

BY

MICHAEL W. LUCAS

CRITICALLY ACCLAIMED
AUTHOR OF
ABSOLUTE OPENBSD,
SSH MASTERY,
AND NETWORK FLOW ANALYSIS

Tilted
Windmill
Press

Sudo Mastery:

User Access Control for Real People

by Michael W Lucas

Tilted Windmill Press

Praise for other books by Michael W Lucas

Absolute OpenBSD, 2nd Edition

"Michael Lucas has done it again." – *cryptednets.org*

"After 13 years of using OpenBSD, I learned something new and useful!" – *Peter Hessler, OpenBSD Journal*

"This is truly an excellent book. It's full of essential material on OpenBSD presented with a sense of humor and an obvious deep knowledge of how this OS works. If you're coming to this book from a Unix background of any kind, you're going to find what you need to quickly become fluent in OpenBSD – both how it works and how to manage it with expertise. I doubt that a better book on OpenBSD could be written." — *Sandra Henry-Stocker, ITWorld.com*

"Do you need this book? If you use OpenBSD, and have not yet achieved guru status, yes, this book is just for you. Even gurus will find valuable things in this book that they did not know… But beyond th OpenBSD aspect, there are great sections on cross-platform applications like *sudo* that are almost enough on their own to justify getting this book. And there are several of those chapters. So: even if you don't use OpenBSD directly, would you like a quick reference on *sudo*, IPv6 networking, and NF setup? Oh, and also *tftpd*, PXE, and diskless BSD systems? But wait, what if I told you these references came with a free book on OpenBSD installation and configuration?" – *Warren Block, wonkity.com*

"It quickly becomes clear that Michael actually uses OpenBSD and is not a hired gun with a set word count to satisfy... In short, this is not a drive-by book and you will not find any hand waving." – *Michael Dexter, callfortesting.org*

DNSSEC Mastery

"When Michael descends on a topic and produces a book, you can expect the result to contain loads o useful information, presented along with humor and real-life anecdotes so you will want to explore th topic in depth on your own systems." — *Peter Hansteen, author of The Book of PF*

"Pick up this book if you want an easy way to dive into DNSSEC." — *psybermonkey.net*

SSH Mastery

"…one of those technical books that you wouldn't keep on your bookshelf. It's one of the books that will have its bindings bent, and many pages bookmarked sitting near the keyboard." — *The Exception Catcher*

"…SSH Mastery is a title that Unix users and system administrators like myself will want to keep within reach…" — *Peter Hansteen, author of The Book of PF*

"This stripping-down of the usual tech-book explanations gives it the immediacy of extended documentation on the Internet. Not the multipage how-to articles used as vehicles for advertising, bu an in-depth presentation from someone who used OpenSSH to do a number of things, and paid attention while doing it." — *DragonFlyBSD Digest*

Network Flow Analysis

"Combining a great writing style with lots of technical info, this book provides a learning experience that's both fun and interesting. Not too many technical books can claim that." — *;login: Magazine, October 2010*

"This book is worth its weight in gold, especially if you have to deal with a shoddy ISP who always blames things on your network." — *Utahcon.com*

"The book is a comparatively quick read and will come in handy when troubleshooting and analyzin network problems." —*Dr. Dobbs*

"Network Flow Analysis is a pick for any library strong in network administration and data management. It's the first to show system administrators how to assess, analyze and debut a network using flow analysis, and comes from one of the best technical writers in the networking and security environments." — *Midwest Book Review*

Absolute FreeBSD, 2nd Edition

"I am happy to say that Michael Lucas is probably the best system administration author I've read. I am amazed that he can communicate top-notch content with a sense of humor, while not offending th reader or sounding stupid. When was the last time you could physically feel yourself getting smarter while reading a book? If you are a beginning to average FreeBSD user, Absolute FreeBSD 2nd Ed (AF2E) will deliver that sensation in spades. Even more advanced users will find plenty to enjoy." — *Richard Bejtlich, CSO, MANDIANT, and TaoSecurity blogger*

"Master practitioner Lucas organizes features and functions to make sense in the development environment, and so provides aid and comfort to new users, novices, and those with significant experience alike." — *SciTech Book News*

"…reads well as the author has a very conversational tone, while giving you more than enough information on the topic at hand. He drops in jokes and honest truths, as if you were talking to him in a bar." — *Technology and Me Blog*

Cisco Routers for the Desperate, 2nd Edition

"If only Cisco Routers for the Desperate had been on my bookshelf a few years ago! It would have definitely saved me many hours of searching for configuration help on my Cisco routers. . . . I would strongly recommend this book for both IT Professionals looking to get started with Cisco routers, as well as anyone who has to deal with a Cisco router from time to time but doesn't have the time or technological know-how to tackle a more in-depth book on the subject." — *Blogcritics Magazine*

"For me, reading this book was like having one of the guys in my company who lives and breathes Cisco sitting down with me for a day and explaining everything I need to know to handle problems or issues likely to come my way. There may be many additional things I could potentially learn about m Cisco switches, but likely few I'm likely to encounter in my environment." — *IT World*

"This really ought to be the book inside every Cisco Router box for the very slim chance things go goofy and help is needed 'right now.'" — *MacCompanion*

Absolute OpenBSD

"My current favorite is Absolute OpenBSD: Unix for the Practical Paranoid by Michael W. Lucas from No Starch Press. Anyone should be able to read this book, download OpenBSD, and get it running as quickly as possible." — *Infoworld*

"I recommend Absolute OpenBSD to all programmers and administrators working with the OpenBSI operating system (OS), or considering it." — *UnixReview*

"Absolute OpenBSD by Michael Lucas is a broad and mostly gentle introduction into the world of th OpenBSD operating system. It is sufficiently complete and deep to give someone new to OpenBSD a solid footing for doing real work and the mental tools for further exploration… The potentially borin topic of systems administration is made very readable and even fun by the light tone that Lucas uses. — *Chris Palmer, President, San Francisco OpenBSD Users Group*

PGP & GPG

"...The World's first user-friendly book on email privacy...unless you're a cryptographer, or never use email, you should read this book." — *Len Sassaman, CodeCon Founder*

"An excellent book that shows the end-user in an easy to read and often entertaining style just about everything they need to know to effectively and properly use PGP and OpenPGP." — *Slashdot*

"PGP & GPG is another excellent book by Michael Lucas. I thoroughly enjoyed his other books due t their content and style. PGP & GPG continues in this fine tradition. If you are trying to learn how to use PGP or GPG, or at least want to ensure you are using them properly, read PGP & GPG." — *TaoSecurity*

Sudo Mastery

Tilted Windmill Press

Sudo Mastery: User Access Control for Real People

Amazon Edition.

Author: Michael W Lucas
Cover design: Bradley K McDevitt
Copyediting: Aidan Julianna "AJ" Powell
Cover Photo: Elizabeth Lucas (concertina wire at abandoned factory, Detroit)

published 2013 by Tilted Windmill Press
www.tiltedwindmillpress.com

The information in this book is provided on an "As Is" basis, without warranty. While every precaution has been taken in the preparation of this work, neither the author nor Tilted Windmill Pres shall have any liability to any person or entity with respect to any loss or damage caused or alleged t be caused directly or indirectly by the information contained in it.

For Liz

Acknowledgements

I want to thank the folks who reviewed the manuscript for Sudo Mastery before publication: Bryan Irvine, JR Aquino, Hugh Brown, and Avigdor Finkelstein. Special thanks are due to Todd Miller, the current primary developer of sudo, who was very patient and helpful when answering my daft questions.

While I appreciate my technical reviewers, no errors in this book are their fault. All errors are my responsibility. Mine, do you hear me? You reviewers want blame for errors? Go make your own.

XKCD fans should note that the author does not particularly enjoy sandwiches. However, Miod Vallat, currently exiled to France, would really like a sandwich with nice fresh bread, really good mustard, and low-carb ground glass and rusty nails. And Bryan Irvine would like a rueben.

This book was written while listening obsessively to Assemblage 23.

Contents

Chapter 1: Introducing sudo

Resolved: controlling user access to a computer's privileged programs and files is a right pain. None of the systems that evolved to cope with mapping real-world privileges onto digital schemes are very good. The best access control systems merely hurt less than others.

Unix-like systems control programs and file access through users and groups. Each individual use has a unique identifier, given either as a username or a user ID number (UID). Users are arranged in uniquely identified groups, given either as a group name or a group ID number (GID). Specific users and groups have permission to access specific files and programs.

This scheme sufficed during UNIX's childhood. A large university might have a couple of UNIX servers. Hundreds of users logged onto each server for mail, news, and computation-intensive applications. Students went in one group, grad students in another, then professors, staff, and so on. Individual classes and departments might have their own groups.

The system owners had a special account, **root**. The **root** account has ultimate system control. As a security and stability precaution, Unix-like systems restrict certain operations so that only **root** can perform them. Only **root** can reconfigure the network, mount new filesystems, and restart programs that attach to privileged network ports. This made sense when you had two servers for an entire campus – reconfiguring the network or adding a new disk drive is a serious task in that environment. The job of managing multimillion-dollar systems should remain in trusted, highly skilled hands.

In the 21st century, Unix-like systems are cheap and plentiful. Teams of people might share systems administration tasks, or one person might have complete control over a system, or anything between. Either situation completely changes your security requirements from those of the previous century.

Large organizations often divide systems administration responsibilities between skilled individuals. One person might be responsible for care and feeding of the operating system, while a second person handles the application running on the server. The server supports the application, and the application is why the server exists, but both people need to perform tasks that require root-level privileges. But root-level privilege is an all-or-nothing affair. There's no division between "access to change the kernel" and "access to run privileged applications." If the application administrator has root-level access, he can change the kernel. You can always rely on gentleman's agreements to only touch the parts of the system you're responsible for, but when your organization employs a team of systems administrators and a team of database administrators to support dozens or hundreds of servers, these gentleman's agreements quickly decompose into finger-pointing bloodbaths – even without vendor-provided application setup scripts that helpfully customize the kernel without telling anyone. These organizations need a finer-grained access control system than **root** provides.

The all-or-nothing model breaks down even more when everyone has a Unix-like system. Setting aside the innumerable phones and tablets which have extra software to make them user-friendly, mar folks run Unix-like operating systems on a desktop or laptop. Every time they access a USB drive or use a coffee shop wireless network, something on the system needs root-level privileges. Using root privileges isn't terribly onerous – log in with your regular account, use the su command to switch users, enter the root password, run the commands that need root access, and exit the root account. Bu when you must use the root account any time you put in a USB drive, bounce the network, add, reconfigure, or restart software, it quickly becomes downright annoying. While software can manage much of this for you, sometimes you must trigger root privileges for routine tasks.

The computing industry is full of really smart people that have expanded the classic UNIX privilege control models. One method is through *setuid* and *setgid* programs. While programs normally work with the privileges of the user who runs them, setuid and setgid programs change thei effective UID and GID to some other value. You can have a setuid program that runs as root. Changing your password requires editing secured files in */etc/*, so the passwd command is setuid. But intruders really like setuid and setgid programs. Flaws in these programs might be exploited into full root access. And most operating systems don't let you make shell scripts setuid, only programs.

Then there are several varieties of *access control lists* (ACLs) which more broadly expand the use group-others ownership model. ACLs allow you to declare something like "This person owns the file but *these* groups and people can modify it, with *these* exclusions, and *these* groups and people (with some exclusions, of course!) can execute it, while these other people can read data from it, except for…" At this point the systems administrator gets a headache and starts contemplating a career cleaning up real sewage instead of the metaphorical kind. And of course, all the different ACL implementations are ever so slightly incompatible. Very few people can correctly implement ACLs c a single platform, and that expertise doesn't really extend to other platforms. ACLs have a place in systems administration, and if you really need them, they're invaluable. But most of us don't need them.

And sadly, access control lists are about as good as it gets.

Except for sudo.

What Is Sudo?

Sudo is a program that controls access to running commands as root or other users. The system owne creates a list of privileged commands that each user can perform. When the user needs to run a command that requires root-level privilege, he asks sudo to run the command for him. Sudo consults i permissions list. If the user has permission to run that command, it runs the command. If the user do not have permission to run the command, sudo tells him so. Running sudo does not require the root password, but rather the user's own password (or some other authentication).

The system administrator can delegate root-level privileges to specific people for very specific tasks without giving out the root password. She can tell sudo to require authentication for some users or commands and not for others. She can permit users access on some machines and not others, all with a single shared configuration file.

Some applications, notably big enterprise database software, run under a specific dedicated account. Users must switch to this account before managing the software. You can configure sudo to permit users to run specific commands as this account. Maybe your junior database administrators only need to run backups, while the lead DBA needs a full-on shell prompt as the database account. Sudo lets you do that.

Finally, sudo logs everything everybody asks it to do. It can even replay the contents of individual sudo sessions, to show you exactly who broke what.

What's Wrong with Sudo?

If sudo is so awesome, why doesn't everybody use it?

Sudo adds another layer of systems administration. Adding that layer requires time, energy, and attention. It requires learning yet another danged program when you already have too much to do. If you're responsible for running an enterprise system with several groups of administrators, investing i sudo reduces your workload. But you must learn how to use it first.

Some commercial UNIXes don't include sudo because they already include their own proprietary escalated privilege management system. OpenSolaris-based systems have `pfexec` and role-based acces control (RBAC). HP has `pbrun`. If you were a commercial UNIX vendor who spent lots of money and energy developing an ACL-based privilege management system, would you include and encourage us of a simpler, easier tool instead? I might, but that's why I'm not a big commercial UNIX vendor.

Many open-source Unix-like operating systems do include sudo in their base system. Some, such a Ubuntu and OS X, completely disable the **root** account and only permit privileged access via sudo. Th is a lurch in the right direction, but most people who have sudo use it incorrectly.

What's the wrong way to use sudo? Sudo is not a replacement for `su`. Sudo is not a way to completely avoid requiring authentication for privileged access. Sudo is not a tool to force someone t make you a sandwich. A proper sudo setup simplifies system management. An improper sudo setup lets intruders and unauthorized users corrupt or destroy your system faster and easier.

"Proper use of sudo" doesn't mean complicated, or even extensive policies. I've seen system administrators spend hours writing complicated sudo policies, only to watch users waltz right past their restrictions. Sometimes the users didn't even realize that the restrictions were in place. Sudo has limits. Once you understand those limits, you can make realistic decisions about how and where your organization deploys sudo.

The problem I see most often with sudo has nothing to do with the software itself. A proper sudo deployment in a complicated organization requires the system administration team to agree who is responsible for what. Sudo enforces job duties and responsibilities in a configuration file. The configuration file is flexible, but people cannot exceed the privileges specified therein.

What are the boundaries of your responsibilities? What permissions do you need to do your real job, and which tasks should someone else do? Being forced to sit down and think about these things can be uncomfortable, and can temporarily increase conflicts within an organization. Once the arguments settle, however, conflicts decrease. There's no bickering over who did what, when, or how Everybody knows that the database team can't format filesystems, the web team can't restart the database, and the sudo logs clearly show who took any privileged actions. And having an audit trail improves system stability. When people know that the system logs their privileged actions, and that they can and will be held responsible for breaking things, they stop breaking things so often. Weird.

Who Does Sudo Protect You From?

Sudo protects the system from harm by intruders or systems administrators, and it protects systems administrators from many management problems.

Giving a user access to only a limited set of privileged commands limits the damage that user can inflict on the system. The user who only has access to manage the web server or database cannot mangle disk partitions. If an intruder compromises that user's account, the intruder is likewise slowe or contained.

Similarly, lack of access protects the system administrator when something goes wrong. Even without sudo logs, a user with limited administrative access can say "Hey, I didn't reconfigure the we server. I don't have that access, remember?" Accountability works both ways. Use it to your advantage.

Sudo Support

Sudo is freely-available open source software. You are welcome to download it from the main web site (http://sudo.ws) or a mirror and use it throughout your organization at no charge. The license permits you to use sudo as the basis of your own products, resell it to clients, or incorporate it into software you then redistribute or resell. You can use sudo for anything you like.

What you don't get is sophisticated support.

Sudo is not created by a commercial company. It's developed and supported by the users who need it, and coordinated for the last several years by Todd Miller. You can contribute to sudo by submittin patches and bug reports. You can find people and companies who will support your sudo install, and who will even write custom code for you. But there's nobody for you to yell at if your sudo install doesn't work the way you expect. There's no toll-free number, no minimum-wage support minion wit a questionable grasp of your language waiting to take abuse and invective in exchange for cash.

Having said that, the people on the sudo mailing lists are both extremely helpful and very interested in real problem reports. They respond well to requests for help and poorly to demands. If you want to demand help – if you want to scream and rant and rave and turn blue in the face until you problem goes away – any number of companies will sell you that.

The software is free. Sudo's "official support" is a gift that evaporates as soon as you stop treating it like one.

Who Should Read This Book?

Everyone who works on a Unix-like system should understand sudo.

If you are a system administrator responsible for maintaining a complicated system, you probably want to assign your application administrators exactly the privileges needed to do their jobs, no more and no less. Correct sudo configuration frees up your time and protects the system from well-intentioned disasters.

If you are an application administrator, you need to do your job. This means you need the access t perform privileged tasks. Working via sudo means changing your processes slightly – not in any major way, but you can go completely bonkers trying to figure out why `sudo cd` doesn't work until you understand what's really happening. An understanding of sudo lets you draft the sudo rules you need and give them to the system administrator. Even if the system administrator disagrees, negotiating in sudo policy language means that you both understand exactly what you're requesting. You can have specific discussions about who is responsible for what. No system administrator will tell an application administrator that he doesn't need the access to manage his application – she certainly doesn't want that job! The only question is: how can that access be best accomplished?

If a disagreement between teams is broad enough, this is where you invoke management to make a very specific decision and set clear lines of authority and responsibility. In some environments, getting that manager to take that step is a miracle in itself. But a mandate to implement sudo lets you corner him. And if you have a cranky system administrator who claims that granting you necessary access without giving you root is impossible, this book will let you categorically refute that. Which, admittedly, is its own vindictive pleasure.

If you maintain only your personal system, why would you care about sudo? Even on a personal laptop, some commands merit more thought and consideration than others. I can understand wanting to trivially reconfigure the laptop's network, tweak removable media, or kill that berserk web browse You probably do these tasks so often that you understand them well – my fingers can configure a network card without disturbing my brain. But tasks you perform less often, such as installing software or formatting disks, require a little more attention. It makes sense to permit sudo to run routine tasks without a password, but to require authentication before upgrading.

Server Prerequisites

This book assumes you're running sudo on a Unix-like operating system. Sudo is available for BSD and Solaris derivatives, Linux, and every commercial UNIX. While my reference platform is FreeBSD, sudo works on all of these systems and more.

My reference implementation is sudo 1.8.8. If you're running an older version, some features migh be absent. A surprising number of operating system vendors include wildly obsolete sudo packages. Check the version of sudo on your system by running `sudo –V`. If your version is much older than 1.8.8 upgrade. You can always get the latest source code and a selection of precompiled packages at the main sudo web site, http:// sudo.ws.

The sudo documentation and this book assume that your operating system conforms fairly closely to the traditional filesystem layout. The examples in this book show commands in standard directorie such as */bin*, */usr/bin*, */sbin*, and so on. If your operating system uses its own directory layout, you'll need to adjust the examples to match.

Sysadmin Background

Where many important programs require an extensive background in related software before you can use them, sudo is nice in that it's fairly self-contained. You can master sudo without understanding al the programs that users can access through sudo. Sudo is a system management tool, however; the more you understand your system, the better you can leverage sudo and the more confidence you'll have in your configuration. I assume you can install sudo, either from an operating system package o from source code.

Configuring sudo requires root access on a Unix-like system and familiarity with a terminal-mode text editor. Sudo defaults to using vi, but you can use Emacs or any other editor.

That's everything. Really. All the other knowledge you'll pick up as we go.