BSD

FreeBSD Tools

DRAGONFLYBSD PACKAGE MANAGEMENT SYSTEM

DATABASE QUERY
OPTIMIZATION
FOR HUGE DATABASES

BRETT DAVIS
OF IXSYSTEMS
TELLS YOU ALL
ABOUT TRUENAS

IS THERE A DOWN SIDE
TO MASS CONNECTIVITY?

FREENAS MINI
STORAGE APPLIANCE

IT *SAVES* YOUR LIFE.

HOW IMPORTANT IS YOUR DATA?

Years of family photos. Your entire music and movie collection. Office documents you've put hours of work into. Backups for every computer you own. We ask again, *how important is your data?*

NOW IMAGINE LOSING IT ALL

Losing one bit - that's all it takes. One single bit, and your file is gone.

The worst part? **You won't know until you absolutely need that file again.**

Example of one-bit corruption

THE SOLUTION

The FreeNAS Mini has emerged as the clear choice to save your digital life. **No other NAS in its class offers ECC (error correcting code) memory and ZFS bitrot protection to ensure data always reaches disk without corruption and *never degrades over time*.**
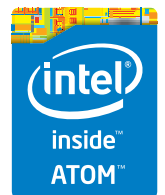
No other NAS combines the inherent data integrity and security of the ZFS filesystem with fast on-disk encryption. No other NAS provides comparable power and flexibility. The FreeNAS Mini is, hands-down, the best home and small office storage appliance you can buy on the market. **When it comes to saving your important data, there simply is no other solution.**

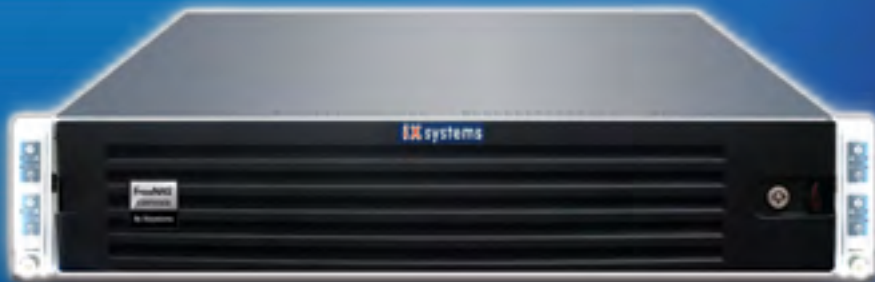**The Mini boasts these state-of-the-art features:**

- 8-core 2.4GHz Intel® Atom™ processor
- Up to 16TB of storage capacity
- 16GB of ECC memory (with the option to upgrade to 32GB)
- 2 x 1 Gigabit network controllers
- Remote management port (IPMI)
- Tool-less design; hot swappable drive trays
- FreeNAS installed and configured

http://www.iXsystems.com/mini

Intel, the Intel logo, Intel Atom and Intel Atom Inside are trademarks of Intel Corporation in the U.S. and/or other countries.

FREENAS CERTIFIED
STORAGE

With over six million downloads, FreeNAS is undisputedly *the* most popular storage operating system in the world.

Sure, you could build your own FreeNAS system: research every hardware option, order all the parts, wait for everything to ship and arrive, vent at customer service because it *hasn't*, and finally build it yourself while hoping everything fits - only to install the software and discover that the system you spent *days* agonizing over **isn't even compatible**. Or...

MAKE IT EASY ON YOURSELF

As the sponsors and lead developers of the FreeNAS project, iXsystems has combined over 20 years of hardware experience with our FreeNAS expertise to bring you FreeNAS Certified Storage. **We make it easy to enjoy all the benefits of FreeNAS without the headache of building, setting up, configuring, and supporting it yourself.** As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS.

Every FreeNAS server we ship is...

» Custom built and optimized for your use case
» Installed, configured, tested, and guaranteed to work out of the box
» Supported by the Silicon Valley team that designed and built it
» Backed by a 3 years parts and labor limited warranty

As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS. **Contact us today for a FREE Risk Elimination Consultation with one of our FreeNAS experts.** Remember, every purchase directly supports the FreeNAS project so we can continue adding features and improvements to the software for years to come. **And really - why would you buy a FreeNAS server from *anyone* else?**

FreeNAS 1U
• Intel® Xeon® Processor E3-1200v2 Family
• Up to 16TB of storage capacity
• 16GB ECC memory (upgradable to 32GB)
• 2 x 10/100/1000 Gigabit Ethernet controllers
• Redundant power supply

FreeNAS 2U
• 2x Intel® Xeon® Processors E5-2600v2 Family
• Up to 48TB of storage capacity
• 32GB ECC memory (upgradable to 128GB)
• 4 x 1GbE Network interface (Onboard) - (Upgradable to 2 x 10 Gigabit Interface)
• Redundant Power Supply

http://www.iXsystems.com/storage/freenas-certified-storage/

Dear Readers,

*J*ust short summing up as it is summer time, and all of us need to have time for some fun.

In the July issue of BSD Magazine, you have a chance to learn how to use Wireshark in a SAN environment. What is more, our expert will teach you how to create a Text Filled Vignette and how to manipulate images like a design professional.

I recommend you read our Interview with Brett Davis. He will tell you more about iXsystems and TrueNAS, the enterprise appliance version of FreeNAS.

Finally, you may find of interest the pkgng tool called "pkg" that is one of the modern and fast binary package managers. This tool was designed and developed for FreeBSD; however, PC-BSD used it in production first and was followed soon after by DragonFly.

I would like to express my gratitude to our experts who contributed to this publication and invite others to cooperate with our magazine.

The next issue of BSD Magazine will be published in 4 weeks. If you are interested in learning more about future content or, if you would like to get in touch with our team, please feel free to send your messages to ewa.d@bsdmag.org. I will be more than happy to answer all your questions.

Ewa Dudzic
and BSD team

FreeNAS
in an Enterprise Environment

By the time you're reading this, FreeNAS has been downloaded more than 5.5 million times. For home users, it's become an indispensable part of their daily lives, akin to the DVR. Meanwhile, all over the world, thousands of businesses universities, and government departments use FreeNAS to build effective storage solutions in myriad applications.

What you will learn...

- How TrueNAS builds off the strong points of the FreeBSD and FreeNAS operating systems
- How TrueNAS meets modern storage challenges for enter

The FreeNAS operating systems is fre
the public and offers thorough doc
active community, and a feature-ric
the storage environment. Based on Free
can share over a host of protocols (SMB
FTP, iSCSI, etc) and features an intuitiv
the ZFS file system, a plug-in system
much more.

Despite the massive popularity
aren't aware of its big brother dut
data in some of the most demand
environments: the proven, enterp
professionally-supported line of
But what makes TrueNAS diffe
Well, I'm glad you asked...

Commercial Grade Supp

When a mission critical stor
organization's whole operat
halt. Whole community-bas
free), it can't always get an
and running in a timely m
responsiveness and expe
dedicated support team
provide that safety.

Created by the sam
developed FreeNAS.

Contents

CYBER SECURITY EXPO®

8-9 October 2014
ExCeL London

A **NEW** event,
for a new era of **cyber threats**

www.cybersec-expo.com

» The most comprehensive analysis anywhere of how to protect the modern organisation from cyber threats

» Free to attend seminars delivered by Mikko Hypponen, Eugene Kaspersky and many more

» Attend the "Hack Den" a live open source security lab to share ideas with White Hat hackers, security gurus, Cyber Security EXPO speakers and fellow professionals

» Network with industry experts and meet with Cyber Security exhibitors

» Discover what the IT Security team of the future will look like

→ Register NOW
www.cybersec-expo.com

Cyber Security EXPO is the new place for everybody wanting to protect their organisation from the increasing commercial threat of cyber attacks. Cyber Security EXPO has been designed to provide CISOs and IT security staff the tools, new thinking and policies to meet the 21st century business cyber security challenge.

Cyber Security EXPO delves into business issues beyond traditional enterprise security products, providing exclusive content on behaviour trends and business continuity. At Cyber Security EXPO, discover how to build trust across the enterprise to securely manage disruptive technologies such as: Cloud, Mobile, Social, Networks, GRC, Analytics, Identity & Access, Data, Encryption and more.

Co-located at

IP EXPO EUROPE
8-9 October 2014 ExCeL London

www.ipexpo.co.uk

FREE REGISTRATION

DragonFlyBSD Package Management System – dports

The recent releases use dports based on the FreeBSD ports collection unlike the older releases which used pkgsrc from the NetBSD project. The pkgng tool called "pkg" is one of the modern and fast binary package managers. This tool was designed and developed for FreeBSD; however, PC-BSD used it in production first and was followed soon after by DragonFly. In the future, it will be the only binary package manager on FreeBSD, just like it is the only port manager in DPorts.

What you will learn…
- What dports is
- How to deal with dports

What you should know…
- FreeBSD Basics

In order to install dports on your system, use the following commands:

```
# cd /usr
# make dports-update
```

If `/usr/local/etc/pkg.conf` exists remove it

```
# rm /usr/local/etc/pkg.conf
```

if `/usr/local/etc/pkg/repos/df-latest.conf` does not exist then create it

```
# cd /usr/local/etc/pkg/repos && mv df-latest.conf.sample
    df-latest.conf
```

In order to install third-party software, you need to use the following commands:

```
# pkg update
# pkg install "software"
```

Working with dports in detail
First if Update pkg repository, search and install a package.

pkg update

Updating repository catalogue

```
digests.txz      100%  989KB 164.8KB/s 278.8KB/s   00:06
packagesite.txz   100% 4372KB 101.7KB/s 261.9KB/s   00:43
```

Incremental update completed, and 21191 packages processed:

20894 packages updated, 139 removed and 230 added.

pkg search nginx

```
nginx-1.6.0,2
nginx-devel-1.7.0
```

pkg install nginx-1.6.0,2
Updating repository catalogue
  The following two packages will be installed:

```
  Upgrading pcre: 8.34 -> 8.34_1 [Avalon]
  Installing nginx: 1.6.0,2 [Avalon]
The installation will require 715 KB more space
1 MB to be downloaded
Proceed with installing packages [y/N]: y
pcre-8.34_1.txz
   100% 1039KB  94.4KB/s  58.8KB/s   00:11
nginx-1.6.0,2.txz
   100%  258KB 128.9KB/s 197.8KB/s   00:02
Checking integrity... done
[1/2] Upgrading pcre from 8.34 to 8.34_1... done
[2/2] Installing nginx-1.6.0,2...===> Creating users and/
  or groups.
Using existing group ,www'.
Using existing user ,www'.
 Now it is done.
```

Removing a package from the System

pkg delete nginx
Uninstallation has been requested for the following packages:

```
  nginx-1.6.0,2
The uninstallation will free 683 KB
Proceed with uninstalling packages [y/N]: y
[1/1] Deleting nginx-1.6.0,2... done
```

You can always delete packages which were automatically installed as dependencies especially if you do not require them anymore.

pkg auto remove
Uninstallation has been requested for the following two packages:

```
  argp-standalone-1.3_2
  mysql55-client-5.5.37
The uninstallation will free 36 MB
Proceed with uninstalling packages [y/N]: y
[1/2] Deleting argp-standalone-1.3_2... done
[2/2] Deleting mysql55-client-5.5.37... done
```

Now, you can list all installed packages on the system and get more details about them.

#pkg info

```
ORBit2-2.14.19          High-performance CORBA ORB
   with support for the C language
Thunar-1.6.3_1          Xfce file manager
appres-1.0.4            Program to list
   application's resources
atk-2.8.0               GNOME accessibility toolkit
   (ATK)
avahi-app-0.6.31_1      Service discovery on a
   local network
bash-4.3.8              The GNU Project's Bourne
   Again Shell
bind98-9.8.7            BIND DNS suite with updated
   DNSSEC and DNS64
bitmap-1.0.7            Bitmap editor and converter
   utilities for X
ca_root_nss-3.15.5      The root certificate bundle
   from the Mozilla Project
cairo-1.10.2_8,2        Vector graphics library
   with cross-device output support
cantarell-fonts-0.0.15  Cantare
-------------- Output Curtailed ------------------
```

2.5: Display current version of 'pkg'.

pkg -v

```
1.2.7_2
```

2.6: Audits installed packages against known vulnerabilities.

pkg audit -F

```
vuln.xml.bz2
   100%  441KB 220.8KB/s  51.5KB/s   00:02
dbus-1.6.18 is vulnerable:
dbus -- local DoS
CVE: CVE-2014-3477
WWW: http://portaudit.FreeBSD.org/52bbc7e8-f13c-11e3-bc09-
   bcaec565249c.html
---------------- Output Curtailed ---------------------
```

Upgrade all packages on the system.

pkg upgrade

```
Updating repository catalogue
```

```
digests.txz
    100%  989KB  27.5KB/s  48.8KB/s  00:36
packagesite.txz
    100% 4372KB  37.7KB/s  11.9KB/s  01:56
Incremental update completed, and 21191 packages processed:
20894 packages updated, 139 removed and 230 added.
Upgrades have been requested for the following 25 packages:

    Installing openssl: 1.0.1_13 [Avalon]
    Upgrading ca_root_nss: 3.15.4 -> 3.16 [Avalon]
    Upgrading curl: 7.35.0 -> 7.36.0_1 [Avalon]
    Upgrading gettext: 0.18.3.1 -> 0.18.3.1_1 [Avalon]
-------- Output Curtailed ------------------------------
    Installing perl5: 5.16.3_10 [Avalon]
    Installing p5-Digest-HMAC: 1.03 [Avalon]
    Upgrading git: 1.8.5.4 -> 1.9.3 [Avalon]


The upgrade will require 131 MB more space
42 MB to be downloaded
Proceed with upgrading packages [y/N]: y
openssl-1.0.1_13.txz
    100% 2542KB  32.2KB/s  21.8KB/s  01:19
ca_root_nss-3.16.txz
    100%  301KB  60.1KB/s  90.7KB/s  00:05
curl-7.36.0_1.txz
    100% 1248KB  33.7KB/s  17.7KB/s  00:37
  ------------ Output Curtailed -----------------------
[25/25] Upgrading git from 1.8.5.4 to 1.9.3...===>
    Creating users and/or groups.
Using existing group 'git_daemon'.
Creating user 'git_daemon' with uid '964'.
Updating /etc/shells
 done
Copy /usr/local/openssl/openssl.cnf.sample to /usr/local/
    openssl/openssl.cnf
and edit it to fit your needs.
---------------- Output Curtailed ----------------------
```

Clean the local cache of fetched remote packages.

pkg clean

The following package files will be deleted from the cache directory

```
/var/cache/pkg:

Package:                  Origin:          Reason:
All/tmux-1.9.a_1.txz      sysutils/tmux    Size mismatch
All/bash-4.3.8.txz          shells/bash
    Superseded by bash-4.3.18_2
---------------- Output Curtailed ----------------------
```

```
All/mime-support-3.54.txz     misc/mime-support
    Checksum mismatch
All/cmdwatch-0.2.0_2.txz      sysutils/cmdwatch
    Checksum mismatch


Proceed with cleaning the cache [y/N]: y
Deleting:
    /var/cache/pkg/All/tmux-1.9.a_1.txz
    /var/cache/pkg/All/bash-4.3.8.txz
    /var/cache/pkg/All/libevent-1.4.14b_3.txz
-----------------Output Curtailed ----------------------
    /var/cache/pkg/All/mime-support-3.54.txz
    /var/cache/pkg/All/cmdwatch-0.2.0_2.txz
All done
```

Conduct sanity checks for installed packages

pkg check -s -a

```
pkg: fopen(/usr/local/etc/smartd.conf.sample): No such file
    or directory
smartmontools-6.2_2: checksum mismatch for /usr/local/etc/
    smartd.conf.sample
```

Query the database for package(s) that installed a specific file

```
# pkg which /usr/local/bin/rsync
/usr/local/bin/rsync was installed by package rsync-3.1.0_3
```

Conclusion

Dports is a versatile package management system. If you want to learn about the options, you need to refer to

man 8 pkg

on your own dragonfly system. Linux users using popular distros will find many similarities in dports with their package management systems such as apt, yum, pacman, emerge, etc.

SIJU GEORGE

Siju George is a software and network engineer. He is working with the National Center of Research (ICAR) area networking in Cosenza as the network manager. He graduated from the University of Calabria, specifically at the DIMES with the "Culture Lab" (http://culture. deis.unical.it) in the Department of Telematics. Currently, He is collaborating with this department on several projects related to mesh networks, and with computer science associations "Hacklab Cosenza" (http://hacklab.cosenzainrete.it/) and "Verde Binario" (http://www. verdebinario.org/).

Getting to Grips With the Gimp – Part 6

A Text Filled Vignette

In the sixth part in our series on the Gimp, we will learn about creating a Text Filled Vignette.

What you will learn…
- How to manipulate images like a design pro

What you should know…
- General PC administration skills

The book, "Tinker Tailor Soldier Spy" by *John Le Carre* has a vignette of George Smiley on the front cover. Hence, we will create a similar edgy picture with text rather than numbers.

The recipe
We will follow the following steps as indicated:

Step 1
Download the image of the female model from Table 1.

Details and Credits	Image	URL
Joann p 02 portrait Uploaded by obyvatel	Female model	*http://www.freeimages.com/photo/668970*

Step 2
Pull the guide down from the top of the measuring bar, so that it rests just beneath the model's right eye [Screenshot 1].
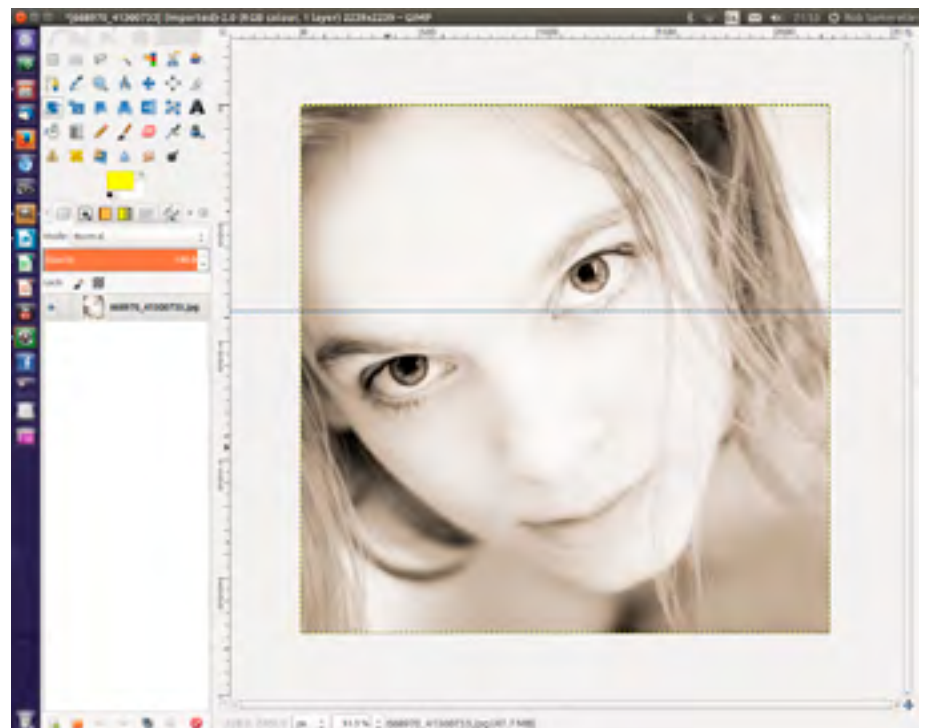
Step 3

Using the *rotate tool*, rotate the image until the eyes become parallel to the guide [Screenshot 2].
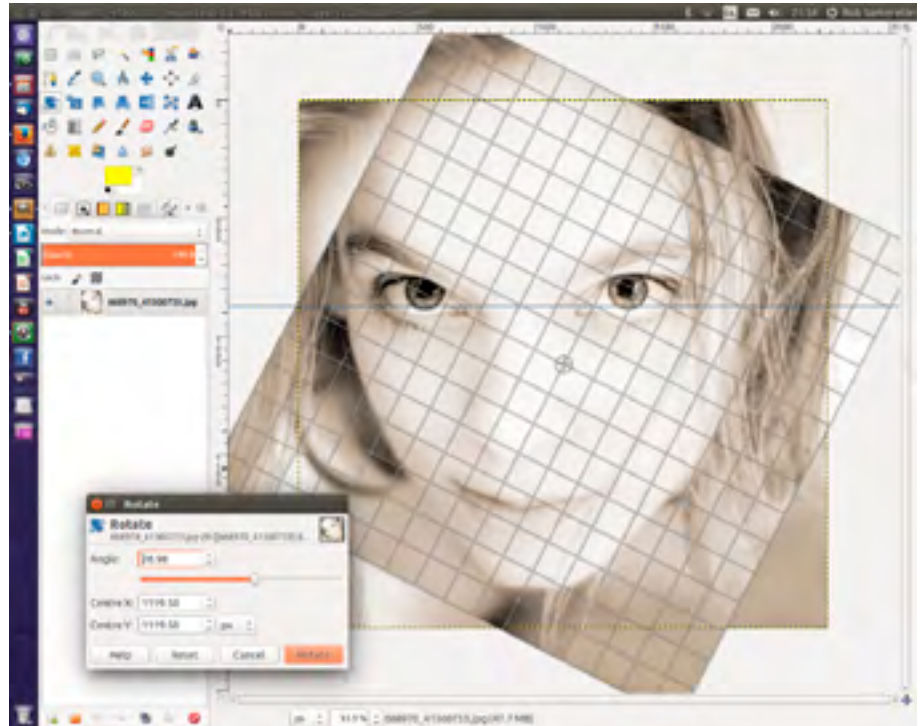
Step 4

Using the *move tool*, bring the layer down until the chin is at the bottom of the frame. Use the crop tool to centre the face by removing excess areas [Screenshot 3].
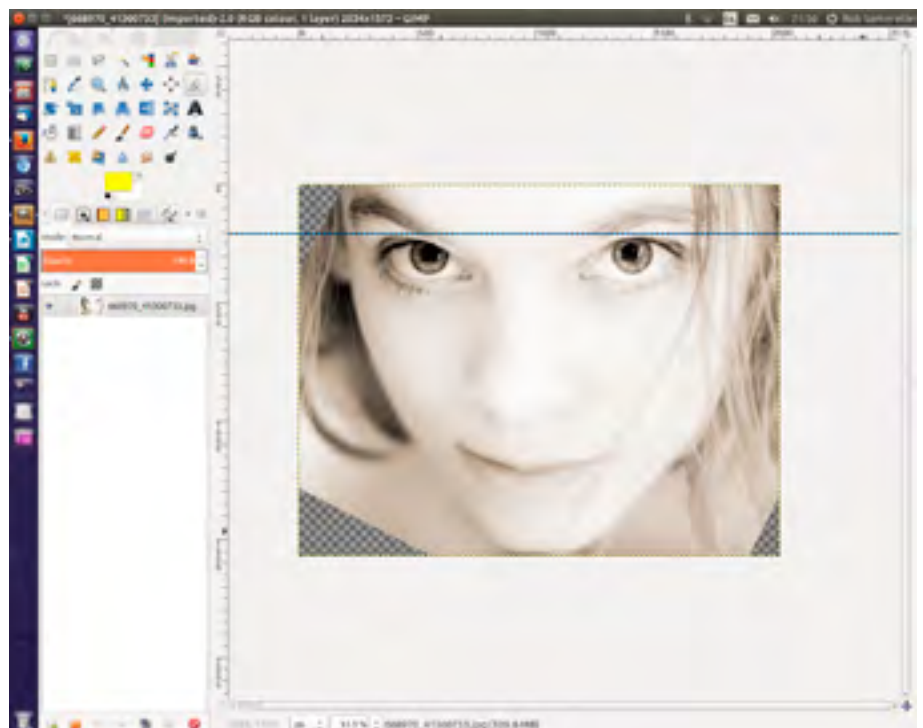
Step 5

Use the *clone tool* to the area of the image left transparent due to the rotation. Use a large size brush to re-create the hair on the left hand side [Screenshot 4].
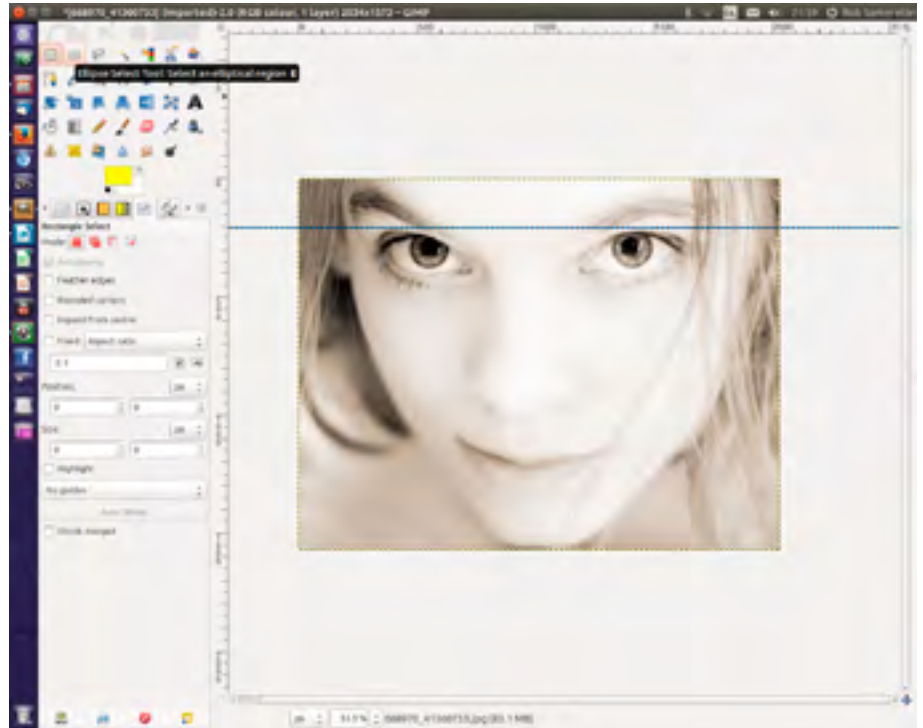
Step 6

Zoom in each eye in turn and carefully select around the iris. Use the shift key to select the area around the second eye. Click on the paths tab and choose selection to path [Screenshot 5-6].
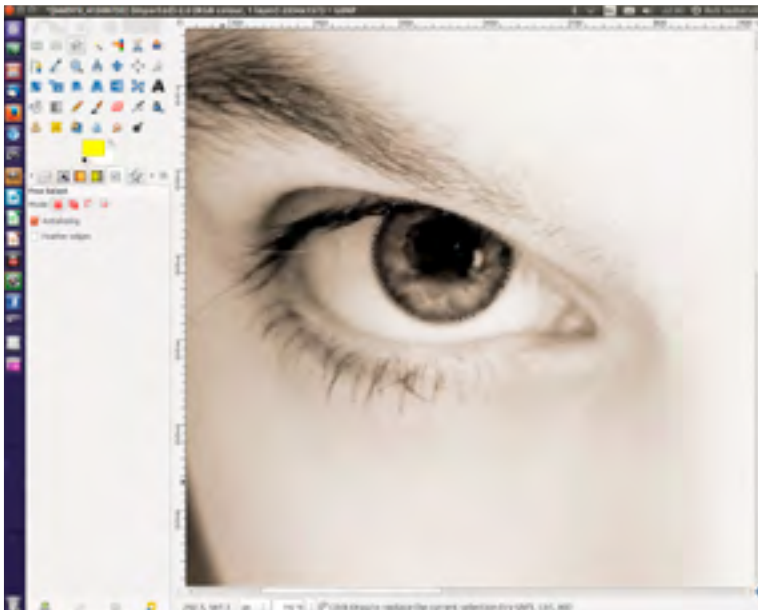
Step 7

Add a new layer, pick a color for the eyes and fill the selected areas. Change the layer mode to *dodge* [Screenshot 7].

Step 8

Using the *smudge tool*, touch up around the eyes dragging the color away from the edge of the iris, so that no sharp edges still remain. Add a new layer and fill with a 45 degree gradient from top left to bottom right. [Screenshot 8].
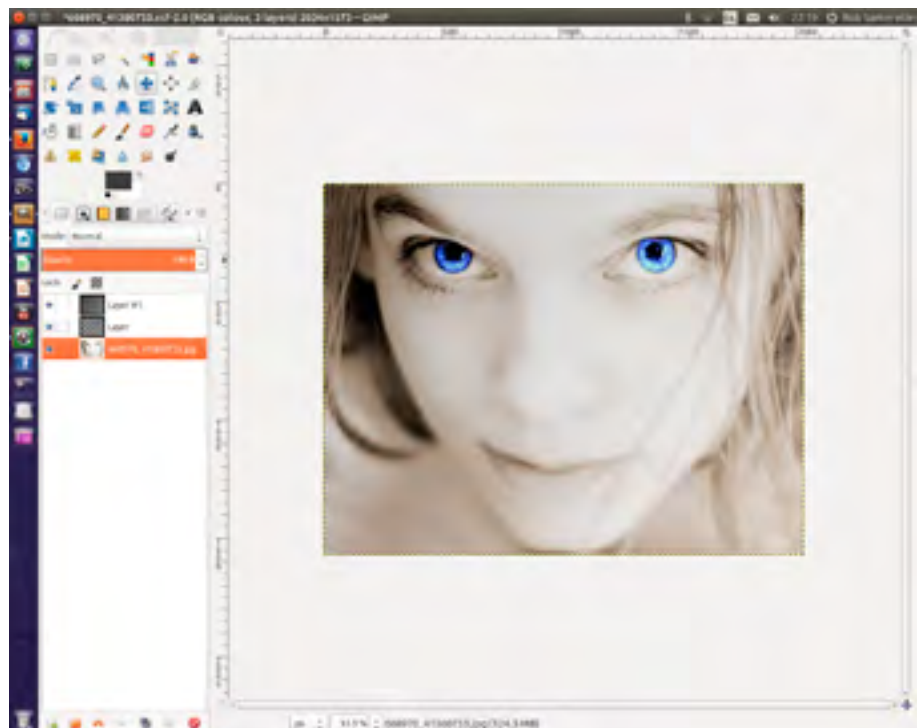
Step 9

Add a new layer as in Step 8, and change the mode to *dissolve*. Adjust the opacity until you get the level of interference you like. Similarly, change the layer in Step 8 until you get an effect you prefer. In the final version, I chose burn [Screenshot 9].
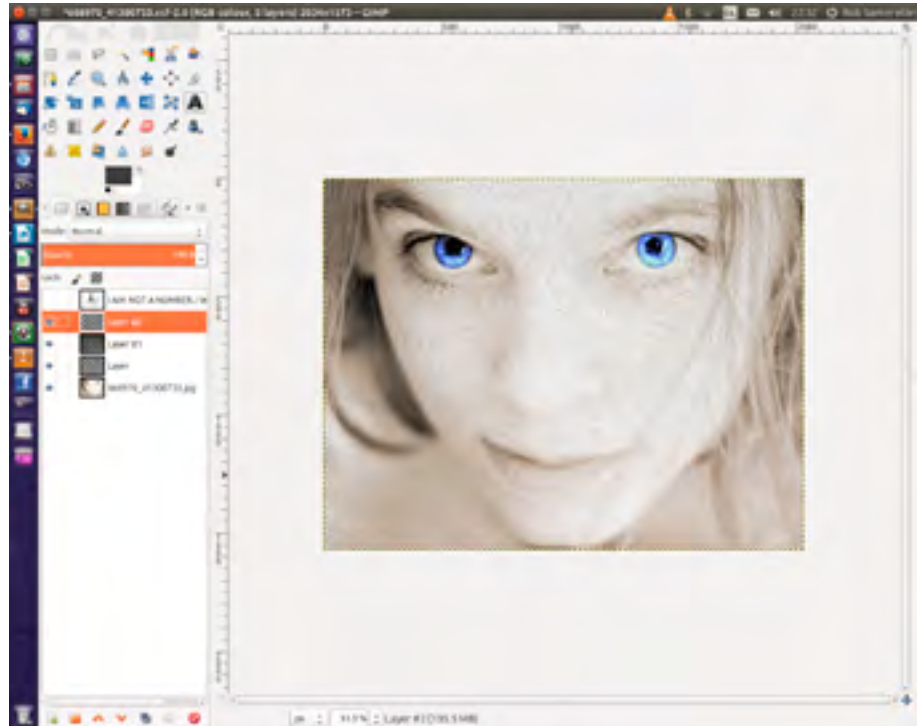
Step 10

Using the *text tool*, select the whole image and then enter the text you want as the *message*. This is very processor and graphics intensive, but Sans 50pt seemed to work OK on my elderly PC. The smaller the font, the more text and the slower this operation will be. Change the layer mode to *overlay mode* [Screenshot 10].
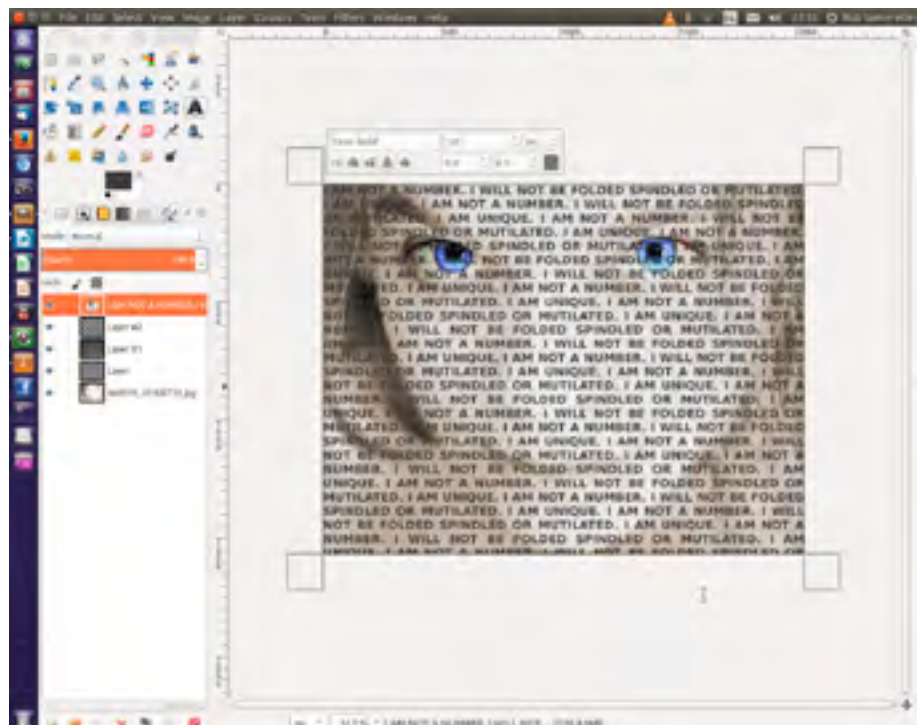
Step 11

Using the *erase tool*, remove all the text detail and gradients from the inside of the eyes. *Experimenting with the text layer*, duplicating it and changing the mode then trying to delete text from the face is a good technique.

Step 12

The final result that I saved will be similar as in [Screenshot 11].

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid-eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

Network Analysis

On a Storage Area Network Using Wireshark

Wireshark, originally known as Ethereal, is probably the most famous open source packet sniffer and network analysis tool available.

What you will learn…
- What Wireshark is
- How to use Wireshark in a SAN environment

What you should know…
- Security Basics

This application supports about 1300 protocols through a vast number of filters. Functionalities such as traffic, protocol analysis, and packet dissector make it an extremely versatile tool for security experts, network engineers, and system administrators.

Wireshark can be used during a proactive analysis to identify potential network bottlenecks, to monitor "live" what is happening to data flow, and to decode packets in transit, displaying information in readable format. The tool can be installed on any computer connected to the network and equipped with a NIC card. Using specific API or libraries, such as WinPcap under Windows or libpcap for Unix, it enables data capture and allows analysis of packets travelling over the carrier.

Commonly, Wireshark is used on Ethernet technology or Wireless networks, but it's also possible to use it for SAN (*Storage Area Network*) to analyze FCP (*Fiber Channel Protocol*) over Optical Fiber Cables.

The Storage Area Network Architecture

SAN (*Storage Area Network*) is generally defined as a dedicated storage network using Fibre Channel technology to provide disk volumes on the target host.

The SAN environment can be designed to have a disk array directly attached to a host or through a SAN Switch (a SAN Network Director similar to an Ethernet Switch) in order to connect multiple hosts to a single array and enable Business Continuity and Disaster Recovery capabilities.

Disks' capacities are presented as logical volumes called LUNs (Logic Unit Number). The provisioning is performed by connecting the Array, Switch and HBA (Host Bus Adapter, a fiber card adapter installed on the Host system) using two different operations called LUN Masking and Zoning (Figure 1).

With Zoning, we connect the ports of the devices, also called initiators, to be logically linked. While performing the LUN Masking, we present the LUN (disk capacity) to the target host.

The SAN directors are accessible by Storage and Network Administrators via the Terminal Access Controller Access-Control System (TACACS) or Remote Authentication Dial In User Service (RADIUS).
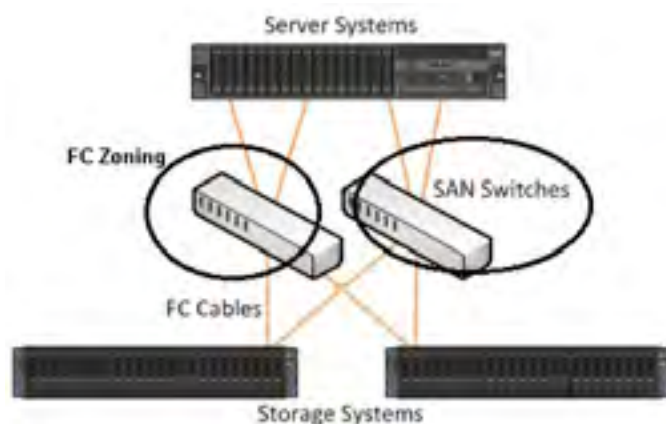
Figure 1. *Fiber Channel Zoning*

The main difference between NAS and SAN volume provisioning systems is the protocol used to provide storage capacity. NAS uses NFS or CIFS protocols while SAN uses the FCP (Fiber Channel Protocol).

Fiber Channel Protocol

The FCP (Fibre Channel Protocol) is a transport protocol similar to TCP/IP, approved as ANSI standard around 1994. FCP mainly transports SCSI commands using the Optical Cable as a carrier (Figure 2). This protocol was invented to enable higher performances and distance insensitivity, to facilitate the system boot from external devices, and to support enterprise storage flexibility and scalability.

Fiber Channel Traffic Analysis

Network analysis on a fiber channel is not the same as over the Ethernet. There's no equivalent promiscuous mode for nodes, so you can't listen to traffic moving through the network. To achieve traffic analysis, first of all, you need to tap into the network between the source and destination ports you wish to analyze. Dedicated hardware is necessary to "read" the packets and specific software to analyze the frames.

Some examples of external frame analyzers are: Xgig Protocol Analyzer Family from JDSU or LeCroy FC Protocol Analyzers.

FC frame analyzers are often accompanied by dedicated TAP (*Traffic Access Point*) network hardware. This device is physically inserted into the network and when turned on, it copies all frames headed for a specific port to a specific TAP port. Using TAP hardware means that the frame analyzer can be plugged into the TAPped port and then removed without causing an interruption in the FC network flow. Of course, in order to initially install the TAP hardware, you have to interrupt the network flow.

Preferably, these devices should be permanently connected because each time you insert and remove the analyzer, you interrupt the FC network flow. This may result in serious repercussions for the system, such as Data Loss and Kernel Panic.

In some cases, this has been made easier by vendors such as Cisco and Brocade, providing a *Switched Port Analyzer* (SPAN) feature, which copies most traffic going to a specific port to another switch port called "mirror port". In that case, the frame analyzer or PAA (*Protocol Analyzer*

Figure 2. *Fiber Cable*
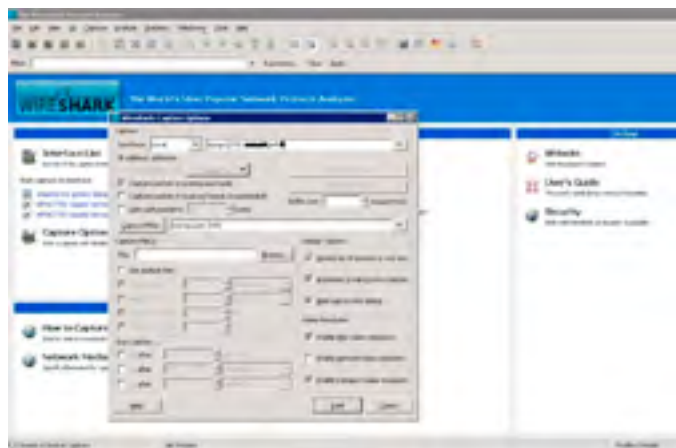
Figure 4. *Setting up Wireshark*
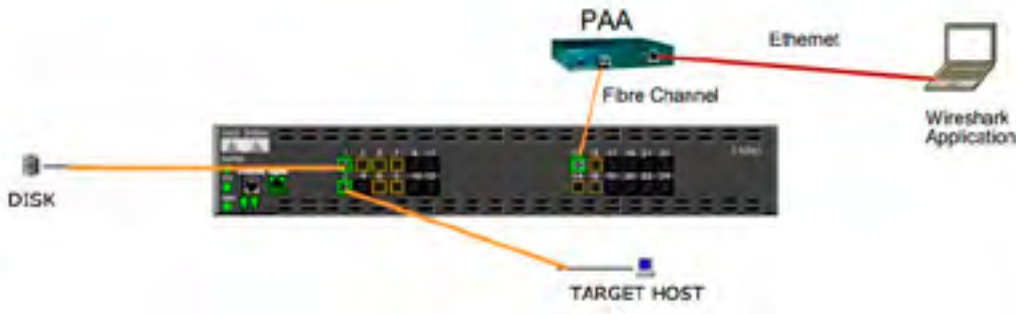
Figure 3. *Typical SPAN to PAA Configuration*

Adapter) can be plugged into the SPAN switch port and analyzes the traffic flow (Figure 3).

Cisco and Brocade provide native command line tools to allow local fiber channel control traffic passing through the local supervisors to be copied into a text file that is stored in a chosen location on the switch or redirected to the IP Address.

The default behavior is to store the output in a volatile storage area. This can later be copied to a remote server for analysis with Wireshark.

It is also possible to specify a remote IP address to send the data to, and Wireshark can be used to analyze the data in real time, as it's collected.

Cisco MDS Switches with the SanOS operating system provide an FC Analyzer command line called: *fcanalyzer* (portlogshow is the command line on brocade).

In order to configure the system to perform traffic analysis, we must configure the Switch in passive remote mode using the command line as follows:

Figure 5. *Remote Connection via Command Line Interface*

Figure 6. *Host Login Trace*

```
MDS3(config)# fcanalyzer remote 172.xxx.xxx.xxx
MDS3(config)# exit
MDS3# show fcanalyzer
PassiveClient = 172.xxx.xxx.xxx
MDS2#
```

Next, we instruct Wireshark to connect to it remotely using the graphical interface (Figure 4). Or, we may try to connect to it using the Wireshark CLI (Figure 5).

Now, we are ready to start a new capture session and verify which type of raw data we can get out of the FC analyzer.

Wireshark can capture a huge amount of information, when installed between the disk array and the host machine. It could potentially intercept all the SCSI commands passing through these two devices. At the same time, it is possible to inspect what is happening at the switch level and use the data for troubleshooting and debugging purposes.

During a live-capture session, we can monitor the Fabric behavior and the Zone-sets operations; or, we can display which initiators and nodes were currently active and enabled. It is possible to verify volumes presented to the hosts and potentially reverse engineer the entire SAN configuration.

We can manage to identify all the Zoning and Masking setup, and if the Switch is using features such as VSAN (Virtual SAN similar to VLAN in Ethernet Networks) or IVR (Inter-VSAN Routing), we can trace all the members' devices existing in all of the SAN areas including all the SCSI command dialogs.

With the help of customized filters, it is possible to use Wireshark for troubleshooting purposes and display (for example, merge conflicts, Fabric Login status, Zoning failure, and so on). A good example is visible in Figure 6. We can see a live capture session with Wireshark tracing a Host Login event. It is possible to trace the entire "dialog" between the Host and the Remote Array through the Switches. There are two active windows in Wireshark:

- Transmit Trace
- Response Trace.

The first one is tracing FCP/SCSI transmission dialog and the second traces the responses.

Appendix 1

- *http://www.cisco.com/en/US/docs/switches/datacenter/mds9000/sw/4_1/configuration/guides/cli_4_1/tsf.html*
- *http://en.wikipedia.org/wiki/Fibre_Channel*
- *http://en.wikipedia.org/wiki/Fibre_Channel_Logins*
- *http://en.wikipedia.org/wiki/Fibre_Channel_zoning*
- *http://www.jdsu.com/en-us/Test-and-Measurement/Products/a-z-product-list/Pages/xgig-protocol-analyzer-family-overview.aspx*
- *http://teledynelecroy.com/protocolanalyzer/protocolstandard.aspx?standardid=5*
- *http://www.brocade.com/products/all/switches/index.page*
- *http://www.cisco.com/en/US/products/hw/ps4159/ps4358/products_configuration_example09186a008026eb55.shtml*

In the first window, we can see LUNs (remote disks) are in "inquiry status" (seeking to log on to target host) and the FC initiator is attempting to initiate the FLOGI (a link service command that sets up a session between two participants' devices).

We can verify the positive response in the second window. The Login request is accepted, and we can see the positive response. The trace window is now displaying that LUNs are reported in good status, hence available to be mounted on the target Host.

Conclusions

This article provides a quick overview of using Wireshark in a SAN environment. Although network analyzers are powerful software and can be used to troubleshoot complicated issues, at the same time they can be extremely dangerous when misused or activated through unauthorized access.

Sniffers are difficult to detect and can be applied almost anywhere within the network under analysis, which makes them one of the hackers' favorite tools.

We need to bear in mind that NO Firewalls or IDS are present in a SAN environment; thus, it is not possible to filter traffic or identify intruders easily.

The Login of a "new" device in the fabric is never reported as malicious activity and poorly monitored. Moreover, a volume can be mounted and shared over multiple hosts and, in most cases, there is no event alert that traces the activity.

It's true that the SAN protocol presents all data at the block level, but it is still possible to capture and dump, in a separate storage area, a large quantity of traffic to attempt file reconstructions later.

Using Wireshark to perform SAN network cartography may be a good starting point to perform further attacks.

One may be able to use the information gathered to reconfigure Zoning and Masking, mount the target volume on a different Host, and access the stored data.

FCP is a protocol that does not provide encryption; thus, all the data travelling is potentially exposed.

Remember to handle all the information gathered with Wireshark carefully in order to avoid data leakage. We should store all the captured files securely, possibly in encrypted volumes and never forget that sniffing is an illegal activity when performed without authorization.

SEMBIANTE MASSIMILIANO

M.S.c. Computer Security Employed at UBS Bank as IT Security and Risk Specialist. Collaborating as Research Engineer at R.I.F.E.C. (Research Institute of Forensic and E-Crimes) focusing on: New Virus, Malware Analysis and reverse, Digital Forensic, Sandbox bypass, Shellcoding, Testing Overflows and Exploitation, Code corruption, Testing unexpected behavior, Privilege Escalation, Cryptography, Cryptanalysis, Data infection analysis, new attack vectors, approaches including new tactics and strategies. Defeating protections, intrusion methodologies, polymorphic and intelligent masquerading. Antivirus adaptation and detection avoidance. Development of Tools and scripts. Web: www.rifec.com | Email: msembiante@rifec.com

Penetration Testing Projects

Memory Traces of the Last 5 Years

In the last five years, I have worked on a series of application penetration testing projects on a very wide variety of products. I have a chance to look at kernel components in Unix environments, antivirus products, storage products, compliance products, mobile products, cloud products, web based products and virtualization products.

What you will learn…
- How to deal with the Unix environment
- How to select the best Pentester's applications

What you should know…
- Security Basics

In the course of my time spent doing penetration testing, I have seen a large number of minor as well as major security issues in these products. In this article, I would like to talk about some of the very interesting security issues that I found doing these pen tests, along with the correct way to fix them.

Please note that all the vulnerabilities being discussed here have already been fixed after the pen test report was submitted (hopefully, there are no regressions in the later versions) and hence, do not expect to find a zero-day in this article. The article is intended to give an idea to the readers about what all can go wrong besides the things shown by the standard pen test tools or fuzzers, and what to look for when you encounter a similar application for the pen test.

Bug 1

Stored XSS from Intranet (CVSS = 7.3) (AV:A/AC:L/Au:N/C:C/I:P/A:P)

It was a bug I found during a penetration test of a scanning server. The architecture dictated that the client needed to send a file name over the LAN, and this scanning server would then let you know the scan status for this file. The activity was logged in the log files of the scanning server. This scanning server had a web management interface, which was accessible by the admin of the scanning server. From the web interface, he/she would be able to see the log files which contained the following information; time-stamp; name of the file; activity log.

Here, the name of the file was something that was externally controlled. Hence, the attack, which would work here, was to send a request for scanning to the scanning server with the file name as a JavaScript code. Now, due to the logging activity, the script would be logged *as is* on the log file since it is treated as a filename. Now, whenever the administrator would log on to the web console, this script would get activated and steal the cookies for the attacker.

This was a persistent XSS attack, but since this required us to send a TCP packet directly to the server, which is positioned inside the network, the range of this attack was Intranet and not Internet, as is the case for most of the other XSS attacks.

I found this attack particularly interesting because, unlike most of the XSS attacks, this was something which didn't need any action on the part of the victim, and it was not entirely a web-based attack. This was a network attack, which targeted the web interface for session hijack.

Bug 2

ActiveX Control Design Flaw (CVSS = 7.8) (AV:N/AC:L/Au:N/C:C/I:N/A:N)

I found this bug while looking at a lab management product. This product was supposed to manage a large number of computers by putting an agent on these machines. Along with the agents, it would also install a bunch of ActiveX controls that was not set with Kill Bits. For those of you who do not know what is a kill bit or ActiveX control, here is a brief description. An ActiveX control can be roughly seen as a COM object, which exposes a bunch of APIs, which are accessible over the web. Any web page can call on these APIs and run the COM object on the local machine. This can act as an entry point from the browser to the actual machine.

Kill bit is a flag, which can be set in the Windows registry. Once set, it disallows the ActiveX control from running on the Internet Explorer or equivalent interfaces. Hence, this is the easiest way to disable the ActiveX control from being called from the web pages.

Coming back to the vulnerability, this ActiveX control had an API called *RunNotepad*. This function would take one parameter, which was intended to be a file name. The effect was that it would open that file in the Notepad application on the local machine. Looking for the possible avenues of attack, I tried fuzzing the string input to this API and calling this API a large number of times. I even tried command injections to start another binary, but this led to nothing. If it had, I would not be writing about this bug here, would I?

Now, looking beyond the obvious, I explored the various command line switches that Notepad could take. It is interesting to see what `/P` could do for notepad. If we run `notepad /P password.conf`, it would redirect the password. conf file to the default printer without any requirement for a user interaction and then it would close the notepad application as well. Hence, this string worked like magic when used as the attack parameter. In our regular day-to-day usage of notepad, we never use switches on this application. It is very interesting to see what all the switches can

do even when the product has been developed correctly with all the checks for buffer overflows, shell metacharacters, format strings, etc.

Bug 3

Setuid Root Binary on Unix (CVSS = 7.1) (AV:A/AC:L/Au:S/C:N/I:C/A:C)

When you are dealing with an appliance or any product where you are pen testing a kernel component, you would typically find a lot of user land binaries which are intended for calling in the kernel functionalities. One of the most commonly found binary types is a setuid binary.

A setuid binary is an executable on Unix which can be called by any user on the machine, but it would always run with the privileges of the owner of the binary. This gets particularly interesting when the owner of the binary file is root itself. Now, when the binary is called, it starts running with root privileges. Since there is a clear privilege escalation, the setuid root binaries are typically self-contained, and they take nothing but the command line parameters as input. This input is very closely sanitized and then consumed in the binary. A very commonly used example of such a binary is the `passwd` command. This binary has to run as root because it has to write to the `/etc/shadow` file in the Unix file system which is root readable only. I was looking at one such setuid binary for a Unix product, and I tried all kinds of command line fuzzing on it. The binary never crashed, and if there were an unexpected parameter, it would safely throw an error message and exit. I must mention here that if you ever see any setuid root binary throwing an error message, this in itself should ring a bell in your head that this may be a vulnerability.

At this point, I would like you to recall the concept of I/O Redirection, which is present in all operating systems, including Unix. Hence, if we redirect the output of this setuid root binary to `/etc/passwd` file, and redirect the `stderr` to `stdout`, this would ensure that the error message would now get written in the `/etc/passwd` file. Please remember that this action would overwrite the passwd file and hence, no user would be able to login into this machine anymore. The exact command to do this would be:

```
$ ./setuid_binary (incorrect_param_to_get_error_message) >
    /etc/passwd 2>&1
```

This issue is a bit tricky to fix because there are bound to be setuid root binaries in the Unix systems, and if

there are incorrect inputs, they are bound to return some error message. The correct way to go about fixing this is to drop privileges to the userid of the caller before writing anything to any open file descriptor, including stdout and stderr.

Bug 4

Authentication Bypass (CVSS = 8.3)
(AV:A/AC:L/Au:N/C:C/I:C/A:C)

One of the biggest mistakes committed by software engineers is to introduce proprietary authentication modules or add proprietary encryption functions with the hope that since the protocol or the algorithm is not known to the general public, they can never break into this. I must emphasize that this is strictly an incorrect assumption. I would quote one such example here where I was supposed to pen test a web application. This web application had a proprietary authentication module over the network that functioned as follows:

• Client would send a TCP Connect Request to the server
• Server would respond with a challenge message after creating a randomly generated file name. For example, it would create a file called `baivinrq9j34fnqer h9nq43invsrb93q4vib9r` in the webroot folder.
• The client is expected to give the name of this file as a response to the server's challenge message.
• When the server receives a response, it will check if the file exists. If yes, it would delete this file and allow access to the client. For the next authentication, it would create another file name; hence, the response would be new.

As you all are correctly thinking, the client can never send the correct response unless and until, the client is on the same machine as the server, and it has access to the webroot folder to figure out the name of this randomly generated file. Well, in the case of this product, this was the case. Hence, the first question to be asked here was – why do we need to open this socket on all interfaces? We could have just opened it on 127.0.0.1 (localhost), if all that was required were an inter-process communication channel over the socket.

This was a trivial flaw compared to what is a significant design flaw in this case. All that the server was doing was to verify the existence of the file whose name was sent by the client as the response message. Hence, if the client wanted, it would have sent the name of any file from the

webroot, and it would have worked. So how about *login. php*?? And yes, it actually worked !!

So, combined with the flow above of opening a socket on all interfaces, this attack became an intranet based attack where anyone could authenticate to the server by connecting to the server and sending the name of any valid file from the webroot. And yes, the cherry on the cake was that the file would get deleted from the webroot, causing a denial of service to the web users.

As an add-on, to make matters worse, the file name here was not restricted to the webroot but it could have been any arbitrary file name in the file system. This server was vulnerable to directory traversal attack as well. Hence the response from the client side could have been `../../../../../../../etc/passwd` and the server would happily delete this file and make the machine inaccessible henceforth.

MRITYUNJAY GAUTAM

Mrityunjay Gautam is a graduate from the Indian Institute of Technology, Kanpur with Computer Science and Engineering as his major. He has several years of experience in the Computer Security industry where he has worked in various aspects of security. His experience includes security design and architecture reviews; security development life cycles; vulnerability assessments; writing exploits; cryptanalysis and vulnerability management. He has multiple national and international publications in the fields of Computer Security and Artificial Intelligence. He also has extensive security training experience in corporate as well as educational institutions.
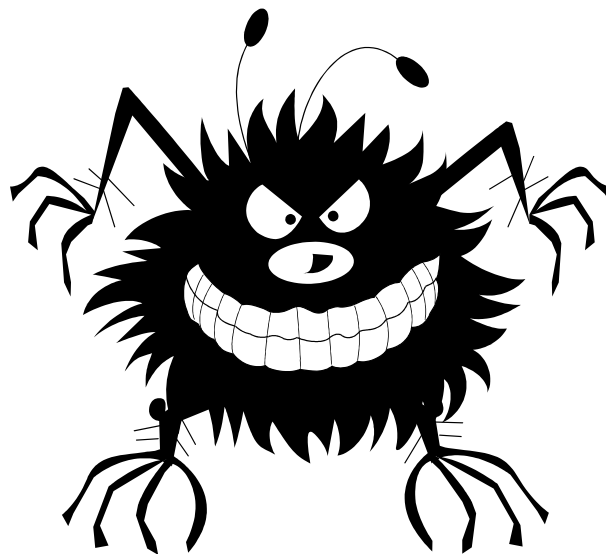
"IN SOME CASES **nipper studio** HAS VIRTUALLY REMOVED the NEED FOR a MANUAL AUDIT"

CISCO SYSTEMS INC.

Titania's award winning Nipper Studio configuration auditing tool is helping security consultants and end-user organizations worldwide improve their network security. Its reports are more detailed than those typically produced by scanners, enabling you to maintain a higher level of vulnerability analysis in the intervals between penetration tests.

Now used in over 45 countries, Nipper Studio provides a thorough, fast & cost effective way to securely audit over 100 different types of network device. The NSA, FBI, DoD & U.S. Treasury already use it, so why not try it for free at www.titania.com

2012
Computing
Security
Awards
WINNER
Enterprise Security
Solution of the Year

Security Products
GOVIES

2012
Computing
Security
Awards
WINNER
Network Security
Solution of the Year

Auto-install Feature Using a Support CD

What you will learn…

- You will learn how to provide answer files for the auto install process.

What you should know…

- You should possess basic UNIX knowledge

First, you need to create your answer file (`autoinstall.conf`); below is an illustrative example,

```
Choose your keyboard layout = fr
System hostname = Box55
Password for root account = hash_password_using_
    encrypt_-b_8
Start ntpd(8) by default = y
What timezone are you in = Indian/Reunion
    Setup a user = oadm
Password for user = hash_password_using_encrypt_-b_8
Location of sets = cd
Set name(s) = -g* -x* +xb*
Directory does not contain SHA256.sig. Continue without
    verification = y
```

This file is hosted by a web server, here a sample: *http://www.mouedine.net/ autoinstall.conf*.

For more details about the answers file, you can read the manual for auto-install (8): man 8 auto-install. If you are ready, it is a good time to start on the support CD ROM (please, order an official CD).

You can see in Figure 1.

Please choose <<`Shell`>> to proceed.

Next step is to enable the Dynamic Host Configuration Protocol (DHCP) for the connected Network Interface Card (NIC) [check installed NICs using "ifconfig" command] as follows:

```
dhclient vic0
```

Now, you can get the answer file using FTP from here: ftp http://www.mouedine.net/autoinstall.conf.

Finally, just type the following command:

```
install -af autoinstall.conf && eject cd0 && reboot
```

WESLEY MOUEDINE ASSABY

Wesley MOUEDINE ASSABY lives in Reunion Island, near Mauritius. He works at AISE-INFORMATIQUE as a network administrator. He has followed OpenBSD closely since 2007 (version 4.1) – it is his passion.

```
root on rd0a swap on rd0b dump on rd0b
erase ^?, werase ^W, kill ^U, intr ^C, status ^T

Welcome to the OpenBSD/i386 5.5 installation program.
(I)nstall, (U)pgrade, (A)utoinstall or (S)hell? s
```
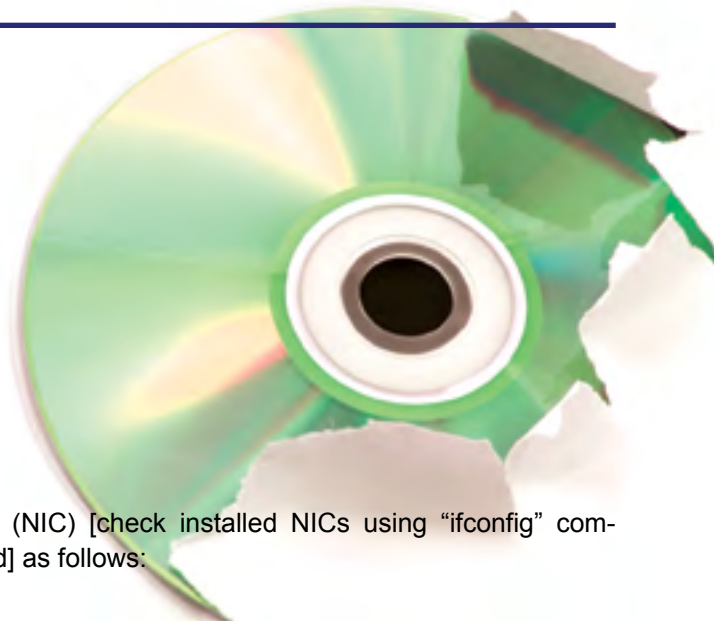
Database Query Optimization for Huge Databases

Occasionally, we have the opportunity to give the database engine a helping hand, and improve the performance of a long-running SQL query. We do this, by not performing the whole query in SQL. We know better than to perform ORDER BY in SQL (see 'Embedded SQL.doc'); now we shall see how to speed up certain types of queries.

What you will learn…
- How to deal with huge databases
- How to speed up certain types of queries

What you should know…
- Database optimization

There are three types of query which benefit from being split up, in this way:

- Any query which performs more than one full scan of a table. This inherently includes all table joins.
- Any query which features more than one logical operation in the predicate. This means any statement beginning with AND, OR and, sometimes, HAVING.
- Occasionally, where the predicate features only one operation, but where this operation is a complex one.

Consider the hypothetical query

```
SELECT x.a, x.b, y.a, y.d
FROM x, y
WHERE x.a = y.a
AND y.d > 42
```

We have joined two tables, and included a numerical comparison, as a filter for our data. The query is, of course, trivial, even if the tables contain millions of rows, but will serve as a useful template, for handling queries with more than one join, and more than one comparison, or other operation.

At the very minimum, the query will perform a full scan of either x, or y, depending on how the optimizer wants to play it, and a full scan of the index of the other table. If there is no index on the columns we've selected, multiple scans of the second table will need to be performed, to find the matching data. Meanwhile, as the data is collected, each row is checked for y.d > 42.

There are two things wrong with this scenario. Firstly, all of the data is read, and re-read from the disk, which is inherently slow; secondly, the join, and the numerical comparison, are both being performed by an SQL interpreter.

So, what can be done, to make the query faster?

We are assuming, as we do with all our examples, that we have an industrial-strength machine, with at least 2GB of RAM. Given this assumption, we will simplify the query, by separating it into two subqueries, which execute as cursors, within our Pro*C, or embedded SQL program.

SELECT x.a, x.b INTO p.a, p.b
FROM x
SELECT y.a, y.d
FROM y

The first query runs to completion, giving us an array of two-element structures, in memory.

Then, the second query runs, through its cursor. As it runs, we perform two operations, which constitute the original predicate.

- We reject any rows where y.d is not greater than 42
- We check the remaining rows, one by one, against our array of structures, for any row, where x.a = y.a. Those which match, we keep; any which don't, we reject.

Why is this faster than allowing the database engine to do it?

The answer is that for our trivial example, it probably isn't. However, if we're extracting those telephone subscribers who live in a particular area, from a table containing twenty million, and joining the result to a table of products, comprising several thousand rows, the difference in run time can be over an order of magnitude.

The key to performing searches on memory-based arrays is a set of tools, the best of which we will now examine.

Hash tables, which we consider first, are an order of magnitude faster than linear searches. However, they are not the fastest way of retrieving data from an array of structures. They carry an overhead of the time taken to actually create the table, which is approximately equal to the time taken to access each element.

A similar disadvantage is shared by the binary tree, or B-tree, described below, which needs time to create the tree, whereas the binary search, or 'divide and conquer' algorithm, requires its input data to be sorted in ascending order.

The difference is that b-trees and binary searches are two orders of magnitude faster than hash tables.

The following summary lists all four methods, applied to retrieving all 2 million elements from an array of 2 million such elements.

The data was originally extracted from a database table, a single scan of which took 87 seconds. This indicates

that it would have taken a prohibitively long time to individually extract each of the 2 million rows.

Table 1. *Methods Summary*

Method	Setup time	Retrieval time
Linear search	0	2200
Hash table	200	200
B-Tree	2	3
Binary search	4	2

Hash Tables

Simplistically speaking, a hash table is a random access matrix of variables and values, where the variables are also the index into the matrix of values.

Instead of saying 'for(I = 0; I < 1000; I++)....', and waiting for the sought after value to fly by, we can simply ask the hash table for the value corresponding to the given variable. Unix has four hash table manipulation functions:

hcreate(length)
allocates space for a hash table, of size 'length' elements.

hsearch(key, ENTER)
Makes a hash table entry, for variable 'key'

hsearch(key, FIND)
Retrieves an entry described by 'key', from the hash table.

hdestroy(void)
Deletes a hash table. The data type of 'key' is defined by the typedef ENTRY, in `<search.h>`, which must be included, if we want to use the hash table functions.

```
struct entry {
    char *key;
    char *data;
};
```

Since hash tables can be used with arrays of complex data structures, the pointers to char are an indication of how the hash table is implemented.

A hash table only stores hash values, and pointers to the original data. This data must persist, with unchanged keys and addresses, throughout the life of the table.

The data itself can change, and the pointer will still correctly retrieve it, but its address must be constant.

For obvious reasons, the key must be unique, so the same criteria must be applied to its choice as are applied to choosing the primary key to a database table. If the target array comprises the rows of a table, which has been extracted into memory, the primary key of the table is an obvious choice.

There is one restriction on the choice of a key. The hash table comparison function is `strcmp()`, which means that the key has to be an ASCII string, and numbers have to be represented by their ASCII values. However, the overhead of the extra `sprintf()` is negligible, compared with the saving of time, especially when scanning a huge array.

By way of example, suppose that we have an array of 1000 data structures, each of which describes a product, as per:

```
struct product {
    char product_code[10];
    char colour[25];
    float length;
    int style;
    float price;
};
struct product products[1000];

ENTRY hhash;
```

Early in our code, we create the 1000 element hash table, like this:

```
 if(hcreate(1000) == 0){
        printf("Can't allocate memory for hash table\n");
        exit(-1);
 }
```

As we create or load the array of product structures, we add an extra step, to make the hash table entries:

```
for(I = 0; I < 1000; I++){

    get_product_data(products[I]);

    hhash.key = (void *)products[I].product_code;
        hhash.data = (void *)&products[I];
        if(hsearch(hhash, ENTER) == NULL){
            printf("Hash table full\n");
        }
```

```
        /* do other stuff… */
}
```

Some time later, we are doing some processing in another loop, and need to find the colour, corresponding to a product code:

```
char * pc;
struct product temp;

for(j = 0; j < 20000000; j++){
    /*
     *     process a lot of customer data
     */
    pc = customer_products[j].product_code;

      /* Now, we need the colour */

    hhash.key = (void *)pc
    if((temp = hsearch(hentry, FIND)) == NULL) {
          printf("No such entry\n");
    } else {
          strcpy(customer_products[I].colour,  ((struct
product *)temp->data)->colour).
    }
}
```

The linear search alternative to using a hash table would have been an inner loop, making up to 1000 iterations, for every iteration of the outer loop.

Binary Trees, or B-Trees

The binary tree is used extensively within many databases, for the creation of indices. Creation of binary trees represents an extremely small overhead, and they are second only to the binary search in terms of data access time, but they provide the added advantage, that nodes can be deleted, and elements of the tree can be modified, while traversing it from a given starting point. The binary search, on the other hand, is just that: a search.

Unix provides a set of binary tree creation and search routines, which have the following functionality:

`tsearch()`	Adds a node to the b-tree
`tfind()`	Searches the b-tree for a given node
`tdelete()`	Deletes a node from the tree
`twalk()`	Traverses the tree, and performs a user-specified action at each node.

To support these activities, we need to supply a comparison routine, identical to the type used by `qsort()`. As with `qsort()`, it needs to take two arguments, which are the nodes to compare, and needs to return –1, 0 or +1, depending on whether the nodes are equal or not. See "Embedded SQL.doc" for code samples.

Syntax for tsearch is:

```
void   *tsearch(void *key, void **root, int (*cmp)(void *,
    void *));
```

The 'key' parameter is a pointer to the element of the structure, by which we will later want to search the tree. The rather ugly **root, is a pointer to a variable, which will contain the address of the root of the tree, when we have one. For the moment, we need to point it at an address of NULL, so we need the following clumsy acrobatics:

```
struct product *xroot;
struct product **root;

    xroot = NULL;      /* this will get set to the root of
  the new tree */
   *root = xroot;        /* this must point to it */
```

Taking the earlier example, using struct product, we would create our b-tree, like this:

```
for(i = 0; i < 1000; i++){
        if(tsearch((void *)products[i].product_code, (void
  **)root, cmpfn) == NULL){
            printf("Error creating binary tree\n");
            break;
        }
}
```

The syntax of the `tfind()` command is identical, except that we don't force a NULL into the address of the root of the tree.

```
void   *tfind(void *key, void **root, int (*cmp)(void *,
    void *));
```

Taking the same example as for the hash table, let's assume that, some time later, we are doing some processing in another loop, and need to find the colour, corresponding to a product code:

```
char * pc;
struct product temp;
```

```
for(j = 0; j < 20000000; j++){
    /*
     *     process a lot of customer data
     */
     pc = products[j].product_code);

       /* Now, we need the colour */

      if(tfind((void *)pc, (void **)root, cmpfn) != NULL){
                printf("Colour >%s< found in tree\n",
  products[i].colour);
        }
}
```

Note that we don't need to perform a loop in order to find our colour.

If we need to delete a particular node, we invoke the `tdelete()` function, with the same syntax as for the preceding commands:

```
void   *tfind(void *key, void **root, int (*cmp)(void *,
    void *));
```

Let 'I' be the index of the array element we wish to delete, then:

```
      if(tdelete((void *)products[I].product_code,
  (void **)root, cmpfn) != NULL){
```

```
        printf("Deleted >%s< from  tree\n",
products[i].product_code);
    }
```

Let us assume that, as part of our data manipulation, we need to update the data in our structures. Perhaps we need to decrease the price for all pink items, because there is a weekend sale.

To perform the equivalent of a global edit, on all the members of the array, individually, would take an extremely long time if, as in real life, the array contained more than the trivial number of items shown above. Worse, if this array had to be processed for every store in the country, inside another loop, the run time would be prohibitive.

This is where we would use `twalk()`, which has a syntax as per:

```
void twalk(void *root, void(*action) (void  *, VISIT,
    int));
```

The parameter 'root' points to the starting node which, in theory, can be any node. However, the traversal is limited to all nodes below this root so, if we wish to visit all nodes, we need to set the root to the first item loaded into our tree, i.e products[0].

The 'action' parameter is a function, which we need to write, to tell `twalk()` what it should do, as it accesses each node. As each node is accessed, `twalk()` will pass to our function, three arguments, of types void *, VISIT, and int. The first argument is a pointer to the node currently being visited, the second is an enumerated data type, whose values are as follows:

0:	preorder.	The node is visited before any of its children.
1:	postorder.	Node is visited after its left child and before its right
2:	endorder.	The node is visited after both children
3:	leaf.	This node is a leaf

Basically, preorder means that this is the first time the node has been visited, postorder means the second time, and endorder means the third time.

Leaf means that this node is a leaf of the tree.

The last argument is the level of the node, relative to the root of the tree, which is level zero.

From these parameters, our function can deduce where we are and what data we're looking at, and perform appropriate actions.

If we had defined a function `modify_data()`, to perform our manipulation, then we would call `twalk()` thus:

```
twalk(root, modify_data);
```

Binary Search

When all we want to do, is find data fast, we use the binary search, which is the fastest of all of the non-linear search techniques. It uses the familiar 'divide and conquer' algorithm, similar to the one used by `qsort()` to sort data.

Before running `bsearch()`, the data must be sorted, using `qsort()`. There is only one function call to remember,

```
bsearch(void *key, void *base, size_t numelmt, size_t
   elmt_size, int (*cmp)(void *, void *));
```

As before, `cmp()` is the usual comparison function, and key is the item for which we are searching. The parameter numelmt, is the length of our array, and elmt_size is sizeof (our data structure). It is called, quite simply, as:

```
char *pc;
struct product *pp;

    pc = products[i].product_code;

    if((pp = bsearch((void *)pc, product, 1000,
  sizeof(struct product), cmp)) != NULL){
        printf("Product >%s< found in array\n",
  pp->product_code);
    }
```

MARK SITKOWSKI

Mark is a Chartered Engineer, and a Corporate Member of the Institution of Electrical Engineers in London. His early career revolved around the writing of analog and digital circuit simulators and digital signal processing applications. In Australia, he moved to writing financial software for the major banks, and telecommunications software for telcos, together with conducting training courses on Unix and database applications. He is currently a consultant to Forticom Security, having written an application for an uncrackable user authentication system.
Design Simulation Systems Ltd: http://www.designsim.com.au
Consultant to Forticom Security: http://www.forticom.com.au
xmarks@exemail.com.au

Inserting and Retrieving Data From Huge Databases

The focal point of our attention, is the database, from which and into which we wish to transfer extremely large quantities of data.

What you will learn…
- How to transfer extremely large quantities of data

What you should know…
- SQL Basics

When we invite our database engine to execute a SELECT statement, two things happen:

- We run our SQL statement through the database server's SQL interpreter
- It performs a read operation on the areas of disk containing the table or tables, which are the subject of our SELECT statement.

If our choice had been a DELETE, UPDATE or INSERT operation, we would have been performing a disk write.

Interpreters are not known for their lightning speed, and disk I/O is the slowest operation that a process can perform. Therefore, we can infer that we should minimize the amount of work done by the SQL, and minimize the amount of reading or writing that the database engine does to the disk. Specifically:

Don't Do The Logic in SQL

Consider a cursor, which is an SQL construct used for extracting multiple rows from a database, executing the following statement on a table of ten million rows:

```
SELECT one, two, three FROM there
WHERE one = 1 AND two = 2 AND three = 3
```

This will execute very slowly, since on every row of the full table scan, which we need to do, we will have to perform three comparisons. It is far more efficient to truncate the predicate to perform only one comparison:

```
SELECT one, two, three FROM there WHERE one = 1
```

and perform the remaining two comparisons in our application:

```
while(sqlca.sqlcode == 0){
    EXEC SQL FETCH cur INTO :one, :two, :three;
    If(two != 2 || three != 3) continue;
    …….
}
```

Never Ask the Database Engine to Do ORDER BY

ORDER BY entails the database engine doing a sort on your data. In order to do the sort, it has to use temporary disk space within its file system. If the database is not idle, but has other users performing operations, and the quantity of data to be sorted is huge, two things will happen:

- You will run out of temporary space
- The operation will take a long, long time.

There is a Unix utility, called `qsort()` which can do, in a few milliseconds, what any database engine can do in a few minutes. Further, it performs an in-place sort, so we don't need huge amounts of temporary space and, far more importantly, we not need disk I/O.

The synopsis of `qsort()` is:

```
void qsort((void *)pointer_to_data,  size_t num_elements,
    size_t sizeof(element), int (*cmp_function)(void *,
    void *);
```

where `cmp_function()` is a user-supplied comparison function for doing the actual sorting. Although the definition looks quite ugly, all it means is that the function has to conform to the following rules:

• It accepts two elements as arguments
• It returns 0 if the elements are equal
• It returns+1 if the first argument is greater than the second
• It returns –1 if the first argument is less than the second.

Now, let us assume that the cursor is executing the statement

```
SELECT one, two, three FROM there WHERE one = 1
```

but we would ideally like it to produce an ordered list, as produced by

```
SELECT one, two, three FROM there WHERE one = 1
ORDER BY one
```

Then, instead of using the `ORDER BY`, we would load the data into an array of structures, which look like

```
struct data {
    int one;
    int two;
    int three;
};
struct data datarray[100000];
```

Then our comparison function would look like this:

```
cmp_data(void *p1, void *p2)                   /* cmp_
    data */


{
```

```
struct data *q1, *q2;

    q1 = (struct data *)p1;
    q2 = (struct data *)p2;
    if(q1->one < q2->one) return(-1);
    else if(q1->one > q2->one) return(1);
    return(0);
}                                       /* cmp_data */
```

and qsort() would be called like: this:

```
qsort((void *)data_p, 100000, sizeof(struct data), cmp_
    data);
```

If we need to ORDER BY more than one variable, we need to adapt the comparison function accordingly. For example, if our original SQL was:

```
SELECT one, two, three FROM there WHERE one = 1
ORDER BY one, two, three
```

we would need the following modification:

```
cmp_data(void *p1, void *p2)                    /* cmp_
    data */

{

struct data *q1, *q2;

    q1 = (struct data *)p1;
    q2 = (struct data *)p2;
    if(q1->one < q2->one) return(-1);
    else if(q1->one > q2->one) return(1);
    else if(q1->one == q2->one){
        if(q1->two < q2->two) return(-1);
        else if(q1->two > q2->two) return(1);
        else if(q1->two == q2->two){
            if(q1->three < q2->three) return(-1);
            else if(q1->three > q2->three) return(1);
            else if(q1->three == q2->three){
                return(0);
             }
        }
    }
}                                       /* cmp_data */
```

If the 'one' elements are equal, we make our decision based on the 'two' elements and, if these latter are equal, we use the 'three' elements.

Run Each Cursor in a Separate Process

It is probable that any application program for high volume data extraction will be using more than one cursor. If this is the case, then we would prefer to run all cursors simultaneously, from separate processes.

In theory, this will reduce the total run time of all of the cursors, to the run time of the slowest.

In practice this depends on a number of factors:

Does The Database Support a 'Dirty Read'?

If the database locks the rows in a table, which is being accessed for any write operation, then our read operation will have to wait for the write to finish.

Oracle will give certain concessions, if we do:

```
EXEC SQL SET TRANSACTION READ ONLY;
```

while Informix explicitly allows:

```
EXEC SQL SET ISOLATION DIRTY READ
```

and DB2 includes isolation level in the cursor definitions.

How Many Other Applications Are Accessing The Same Tables?

Obviously, if we have only one process, accessing the area of disk containing our tables, we will be able to achieve a higher read rate. The above notwithstanding, running multiple cursors from separate processes will always lead to a performance advantage.

On a single-processor machine, the separate processes will each occupy a separate slot in the process table. If we have 'n' cursors, this will give our application 'n' times the CPU time of a single process.

On a multi-processor machine, in addition to the above advantage, we will almost certainly get a processor for each cursor.

Use a PREPARE Statement for Cursors

To avoid duplicating the boilerplate code that creates a cursor, it is more convenient to define the SQL in a character string, and to pass it in to one function, which makes a generic cursor.

```
char *astring = "SELECT one, two, three FROM there \
                WHERE one = 1 AND two = 2 AND
    three = 3";

char *bstring = "SELECT four, five, six FROM here
```

```
                          WHERE four != 4 AND five != 5 AND
    six != 6";
/*
         *       We can then call the function below, like
    this:
 *                   cursors(astring, 1);
         *                   cursors(bstring, 2);
 */

cursors(char *curstring, int which)          / *
    cursors */

{

   switch((pid = fork())){
        case -1:
              printf("Fork from cursors() (%d) failed\n",
    which);
              perror("Fork");
              exit(-1);
        break;
        case 0:                              /* this
    is the child process */
              if(setpgrp() == -1){
                   printf("Warning: Cursor Child can't set
pgrp\n");
              }

              /* Connect to DB */
              EXEC SQL
                 CONNECT :user IDENTIFIED BY :pswd AT DB_
    XYZ  using :dbase;

              if(sqlca.sqlcode != 0){
                   printf("PID %d:Connection refused:%s\n",
    getpid(),

    sqlca.sqlerrm.sqlerrmc);
                   exit(-1);
              }
              printf("PID %d (%d):Connected to DB %s\n",
    getpid(), which, dbase);

              EXEC SQL AT DB_XYZ SET TRANSACTION READ ONLY;

               /* Prepare the cursor from the incoming
    string */

              EXEC SQL AT DB_XYZ PREPARE xcur FROM
    :curstring;
              if(sqlca.sqlcode != 0){
```

```
              printf("PID %d:Can't prepare cursor:%s\n",
getpid(),

sqlca.sqlerrm.sqlerrmc);
              EXEC SQL AT DB_XYZ  ROLLBACK WORK RELEASE;
              exit(-1);
       }

       /* declare the cursor */
       EXEC SQL AT DB_XYZ DECLARE gcur CURSOR FOR
xcur;
       if(sqlca.sqlcode != 0){
              printf("Can't declare cursor:%s\n", sqlca.
sqlerrm.sqlerrmc);
              EXEC SQL AT DB_XYZ ROLLBACK WORK RELEASE;
              exit(-1);
       }
       /* Now open it */
       EXEC SQL AT DB_XYZ OPEN gcur;
       if(sqlca.sqlcode != 0){
              printf("Can't open cursor:%s\n", sqlca.
sqlerrm.sqlerrmc);
              EXEC SQL AT DB_XYZ ROLLBACK WORK RELEASE;
              exit(-1);
       }
        /* Now, the individual cursor-specific code */
       switch(which){
           case 1:
             while(!(sqlca.sqlcode == 1403 || sqlca.
sqlcode == 100)){
                      EXEC SQL AT DB_XYZ
                       FETCH gccur INTO :one, :two, :three;
              }
             break;
             case 2:
                  /* fetch cursor 2 */
             break;
             case 3:
                  /* fetch cursor 3 */
             break;
             etc….
       }
 default:                               /* back in
the parent process */
    printf("Child process %d running\n", pid);
  break;

}

/* cursors */
```

Use the Oracle Array Cursors for Input or Output

Oracle has the facility of fetching a cursor into a host variable which is an array.

The advantages of this are considerable. With one SQL request, we can fetch, not just one row, but any number, like 20000, or 30000. Equally, we can perform an INSERT from an array host variable, and load 50000 rows, or however many we feel should form our syncpoint.

The Array FETCH is Performed Like This

Declare all input host variables as arrays:

```
#define FCH 32000

struct  data{
    int one[FCH];
    int two[FCH];
    int three[FCH];
};
```

Define the cursor string:

```
char *astring = "SELECT one, two, three FROM there \
                        WHERE one = 1 AND two = 2 AND
   three = 3";
```

- PREPARE, DECLARE and OPEN the cursor, as described above.
- FETCH the cursor into the host variables.

This is a little more complicated than a usual FETCH, since it is unlikely that the size of our array divides exactly into the number of rows in the table. We need to know how many rows were returned, with each iteration of the cursor.

This is achieved by using the sqlerrd[2] member of the sqlca structure, which holds a running total of the rows returned. For convenience, we define two local variables:

```
int prev_fetch;                /* the number of rows we
   got last time */
int current_fetch;             /* the number of rows in
   this fetch */
```

We then FETCH the cursor into our array variables:

```
while(!(sqlca.sqlcode == 1403 || sqlca.sqlcode == 100)){
                EXEC SQL AT DB_STG FETCH gccur INTO
   :data.one,
```

```
   :data.two,

   :data.three;

/* 32000 in one FETCH */
                if(sqlca.sqlcode != 0){
                    if(sqlca.sqlcode != -1405){  /*
ignore fetched NULL */
                        EXEC SQL AT DB_XYZ  CLOSE
gccur;
                        if(sqlca.sqlcode != 0){
                            printf("Can't close
cursor:%s\n", sqlca.sqlerrm.sqlerrmc);
                            EXEC SQL AT DB_XYZ
ROLLBACK WORK RELEASE;
                            exit(-1);
                        }
                        break;
                    }
                }
                if(sqlca.sqlerrd[2] == 0) break; /* no
data */
                /*
                 * sqlca.sqlerrd[2] is a running
total, so we must
                 * subtract the previous total, to get
the current
                 * number fetched
                 */
                current_fetch = sqlca.sqlerrd[2] -
prev_fetch;

                prev_fetch = sqlca.sqlerrd[2];
                if(current_fetch == 0) break;
}
```

MARK SITKOWSKI

Mark is a Chartered Engineer, and a Corporate Member of the Institution of Electrical Engineers in London. His early career revolved around the writing of analog and digital circuit simulators and digital signal processing applications. In Australia, he moved to writing financial software for the major banks, and telecommunications software for telcos, together with conducting training courses on Unix and database applications. He is currently a consultant to Forticom Security, having written an application for an uncrackable user authentication system.

Design Simulation Systems Ltd: http://www.designsim.com.au
Consultant to Forticom Security: http://www.forticom.com.au
xmarks@exemail.com.au

NET OPEN SERVICES IS AN APPLICATION HOSTING COMPANY FOCUSED
ON OPEN SOURCE APPLICATIONS MANAGEMENT IN HIGH AVAILABILITY ENVIRONMENT.

NET OPEN SERVICES IS PROUD TO PROVIDE A HIGH QUALITY SERVICE TO OUR CUSTOMERS SINCE 10 YEARS.

OUR EXPERTISE INCLUDES:

- CLOUD COMPUTING, PUBLIC, PRIVATE AND HYBRID CLOUD MANAGEMENT
  (OPENSTACK, CLOUDSTACK, RED HAT ENTERPRISE VIRTUALIZATION)

- REMOTE MONITORING AND MANAGEMENT 24/7

- NETWORKING AND SECURITY
  (OPEN BSD, IP TABLE, CHECKPOINT, CISCO,...)

- OS AND APPLICATION MANAGEMENT
  (FREE BSD, OPEN BSD, SOLARIS, UNIX, LINUX, AIX, MS WINDOWS)

- DATABASE MANAGEMENT
  (ORACLE, MYSQL, CASSANDRA, NOSQL, MS SQL, SYBASE...)

- MANAGED HOSTING IN CARRIER CLASS DATA CENTERS

- DISASTER RECOVERY

WE PROVIDE SERVICES IN EVERY STEP OF THE PROJECT LIFE, DESIGN, DEPLOYMENT, MANAGEMENT AND EVOLUTIONS.
NETOPENSERVICES TEAM INCLUDES EXPERIENCED LEADERS AND ENGINEERS IN THE INTERNET SERVER INDUSTRY.

OUR TEAM HAS **15 YEARS OF EXPERIENCE** IN DEVELOPING INTERNET INFRASTRUCTURE-GRADE SOLUTIONS AND PROVISIONING INTERNET
DATACENTERS AND GLOBAL SERVICE NETWORKS TOGETHER.

WE OFFER EXCEPTIONAL HARDWARE SUPPORT AS SOFTWARE SUPPORT ON UNIX/LINUX AND OPEN SOURCE APPLICATION.
NETOPENSERVICES DELIVERS THESE CUSTOM-BUILT LINUX AND UNIX SERVERS, AS WELL AS PRECONFIGURED SERVERS AND SCALABLE STORAGE
SOLUTIONS, TO OUR CUSTOMERS. WE ALSO OFFER CUSTOM DEVELOPMENT AND ADVANCED-LEVEL UNIX/LINUX CONSULTING SOLUTIONS.

WWW.NETOPENSERVICES.COM • CONTACT@NETOPENSERVICES.COM

Interview with Brett Davis

Can you please introduce yourself and explain when and how you got in touch with iXSystems and TrueNAS?

This year marks 10 years at iX for me. I joined in late 2003 as the first and only dedicated salesperson and fourth employee overall. At the time, the company had been around in its current iteration for a little less than a year, following an employee asset buyout after the WindRiver acquisition of BSDi. It was a dingy little office. No one was taking salaries. It was a pretty typical post "dot-bomb" era startup scene. What wasn't typical, though, were the uniqueness of the personalities and zealous passion for Open Source software, most specifically, *BSD. "Making money" had been the driving force for any and every other for-profit company I'd worked for, but I immediately realized that the priorities at iXsystems were different. Not that they didn't want to make money, but rather, money was always discussed as just a tangible side benefit of doing something they really cared about (furthering BSD), not a primary focus. It was an extremely refreshing and exciting energy then, and we've been successful at keeping that intact some 10 years and many employees later.

TrueNAS is a good example of this, too. For those that may not know, TrueNAS is essentially the enterprise appliance version of FreeNAS. And, for those that also may not know, iXsystems is the corporate sponsor of the FreeNAS Project (*www.freenas.org*), which is now the world's most popular storage OS. When taking over development of that project in 2010, our main objective was to keep FreeNAS a part of the *BSD family, after some talk in the community about refactoring the code and basing it on Debian. There was never discussion of closing the source or making it paid software. The goal at hand was to keep it BSD and continue to improve it for all the world to use. This was misunderstood by some at first. There was some talk of iX being the "evil corporate overlord" with nefarious plans to deprive the world of this awesome, free software. We never let that bother us; we found it kind of funny, actually. Instead, we chose to speak through action, rewriting the software from scratch to allow the inclusion of ZFS and keep pace with modern FreeBSD, and we now have a 4+ year track record of continuing to improve it. And, it's still free and always will be.

But, throughout the process of re-writing FreeNAS, we learned a lot, and one of the most interesting discoveries was how widely FreeNAS was being used. We were shocked at some of the massive enterprise deployments we were finding out about. Although, the way we were finding out about them was usually pretty tragic. It would typically go something like this: we'd receive a desperate support request from a System Administrator at XYZ Fortune 500 company that had built a massive infrastructure out of some old Dell servers, or had read an article about the Backblaze and was brave enough to try and build one for enterprise use, and had some sort of outage. After trying to help some of these folks, most of the time in vain, spending hours trying to piece together in our minds the hardware abomination they'd built before we could even then determine if we were looking at a software or hardware issue, we realized something: storage is hard. More specifically, "Software Defined Storage" is hard.

There are so many moving pieces and endless hardware combinations that if you're building anything more than a simple file server, this notion of "just load this software onto any old hardware you have lying around, and you can replace your NetApp" is really kind of a myth and a dangerous game for businesses to be playing. Software Defined Networking on the other hand? That I get because you're basically just buying an embedded whitebox switch and putting software on it. The hardware is simple, and you can save some money. But, with storage, there are just too many variables. So, we realized pretty quickly that an enterprise appliance version of FreeNAS was necessary so that businesses could receive the kind of reliability, predictability, and enterprise-class support that is required for something as critical for businesses as their data. We could have tried to ride the whole "software defined" wave. It would have been the easy cash grab but would have gone against the very principles by which the company was founded. So, through the experience of trying to support FreeNAS, we realized it was actually the opposite of what businesses needed. And, that's how the TrueNAS line of appliances came to be back in 2011.

What are the main innovations of the new TrueNAS models compared to the previous generation?

We're pretty proud of this new TrueNAS lineup, released in July. The first thing you'll notice is that the product line is slimmed down quite a bit. We've achieved this by designing a purpose-built, modular hardware platform that is the foundation for all three versions of the appliance. This design makes the appliances easier to manage and maintain – everything is tool-less and swappable. Every model is also now upgradeable to "high availability" by adding a second storage controller, something that wasn't possible with the last generation. Under the hood you'll find more powerful hardware across the board: faster CPUs with more cores, more RAM, faster controllers, better chipsets. Not to say the previous generation was lacking in those departments, but the hardware is just more modern now. I'm also told it's much sexier, which is probably more important than any of us would like to admit.

What are the main advantages of TrueNAS?

TrueNAS' advantages over FreeNAS are simple: TrueNAS is a full blown appliance, specifically designed for enterprise storage and all its trappings. FreeNAS is just software. There are also some minor software differences tied to the specific hardware we use for TrueNAS as well: high availability (failover), performance tuning,

drive failure LED notification, global fault light indication on the bezel (TrueNAS logo turns from white to red when there are system alerts), specific enclosure management hooks, hot spare drives, etc. Exposing these features to the potentially infinite number of FreeNAS builds just isn't possible.

But, even more important than those differences is the comprehensive support available for TrueNAS. Even as the creators of FreeNAS, we cannot provide the quality of support for FreeNAS that mission critical environments require. It just isn't scalable, and we're not interested in providing something that isn't of consistently high quality. We've watched other Software Defined Storage companies try, gain some success, but ultimately fall flat once enough people catch wind of the difficult and painful customer experience when things go wrong. TrueNAS allows us to avoid that mess entirely and make sure our customers have the peace of mind that if something does go South, we're always here to provide a prompt resolution.

TrueNAS is also part of the VMWare Ready and Citrix Verified programs, meaning that it's an officially supported platform for virtualized environments. This isn't something we can do with the FreeNAS software, again due to the potential variability of the hardware.

When compared to other traditional storage appliances, TrueNAS' advantages are in the power of the ZFS file system as well as the open source development methodology. I won't belabor the ZFS point, since that's a known quantity around these parts, but the open source model gives us a distinct advantage over our competition, which is our accelerated development pace. FreeNAS has a pretty decent sized community and some very engaged community members. This allows our alpha and beta cycles to be shorter due to the constant stream of collaborative testing between our developers and the community. No QA team could work this efficiently. This model blesses us with the agility to add new technologies and features at a much faster clip than our big, slow-moving competition, while giving us extra assurance the software is solid. TrueNAS then follows behind FreeNAS by a few weeks, just to allow for additional soak time and for our own internal QA process.

Where do you see the product line growing in the near future?

We have a lot planned for TrueNAS on the software side of things that are now made possible by the streamlined

product line, most of which are related to continuing to make the TrueNAS customer experience the best in the industry. We're looking at predictive drive failure notification, improvements to the support and reporting modules, and other tweaks that make things even easier to use and manage.

Plugins are currently only available in FreeNAS because most of them are designed for home use (PLEX, owncloud, sickbeard, etc), but we have plans for some enterprise-specific plugins on the horizon. We are also going to be adding the support of a few more protocols as well. Extending our VMWare integration by becoming a certified VAAI platform is also on our near-term roadmap. These are a few things coming up that I'm at liberty to discuss.

Who do you see TrueNAS competing with? Synology/Readynas? EMC/Netapp?

The power of the ZFS file system automatically puts True-NAS (and even FreeNAS) in a class apart from the SOHO NAS manufacturers you mention. ZFS is among the elite echelon of storage file systems. If you're using a general purpose storage file system like EXT3/4 or XFS for storage, or trying your luck with something as nascent and untested as ZFS on Linux, then you obviously don't really care about your data. That, or you don't know any better.

We find that we're mostly up against the big guys you mention; replacing legacy installs and providing most of the same functionality, and similar if not better performance, typically with one less zero in the price tag.

We also see some competition from the Software Defined Storage players here and there, though that trend really seems to be on a downward arc over the last two years.

Will TrueNAS ever be available as a standalone software license?

We get asked this a lot, but the answer is no. We believe that the jig is up for Software Defined Storage's utility in enterprise IT environments now that enough people are discovering that the cost savings are mythical. It's our opinion that offering TrueNAS as a standalone software to install on any commodity hardware would be a short-sighted move that would contrast against the best interest of our customers.

FreeNAS software, however, will always remain available for home NAS and small offices or simple, non-criti-

cal file servers, and/or those willing to support themselves with the help of the community.

How is the VAAI (VMware primitives) integration going?

From a development perspective, it's going quite well. I believe we'll have it unofficially available soon, but then we have the hard part: the VMWare certification process. My hand is already starting to cramp thinking of all the checks I have to write.

Will iXsystems integrate flash storage (whether PCI-Express or SSD) into upcoming TrueNAS product lines?

SSD flash storage is integrated throughout the line as ZIL and L2ARC, and always has been. SSDs are also available for primary storage as well. We have also done PCI-Express flash for L2ARC in the past, however we've moved away from it since it is problematic for failover.

Are there plans to provide programmatic management and configuration of TrueNAS devices, like with chef or puppet or ansible etc.

There are, but I'll have to defer to Jordan Hubbard for details on that one ;-)

Summing up, please tell our Readers why TrueNAS is so unique and what the company can achieve when they decide to use it?

TrueNAS essentially enables FreeNAS & ZFS to be utilized in a business environment with the confidence in knowing that the hardware is qualified and purpose-built for the software, tuned appropriately, and there are folks always waiting by the phone to assist when needed.

It's a powerfully-flexible and scalable line of appliances that easily integrates with heterogeneous storage environments at nearly every tier. What that means is that there aren't many storage requirements that can't be solved with TrueNAS. And, the ones that can't are being worked on in the lab, I assure you.

And, lastly, people can feel good knowing that an investment in TrueNAS (or any iXsystems product for that matter) is an investment in the thing we're all here for, after all: the continued development and proliferation of all things *BSD.

BY LUCA FERRARI

Meet the
Developer-Friendly
Payment Solution

Conversions

Payment page

Payment flow

3 easy steps to optimized checkouts:

1

Create the checkout page

With Gate2Shop, you can optimize your payment pages by using ready-made templates or by customizing payment pages to your site look and feel.

2

Test and optimize

An effective payment page variant testing tool, A/B Testing helps you gain insight into user behaviour, increase payment conversion in the short and long term.

3

Accept payments worldwide

With dozens of alternative and local payment methods offered in multiple currencies, the personalized checkout allows you to reach users from all around the world.

✔ Easy integration   ✔ Cross-platform   ✔ Secure

G2S gate2shop
Sell. More.

Call for a free consultation: +44 20 3051 0330
www.g2s.com

While it cannot be disputed that the World Wide Web and the Internet has helped in speeding up the advancement of globalization in eroding national barriers – is there a down side to mass connectivity?

I have just finished reading an excellent book by *Alex Perry* – the award winning Time journalist – entitled "Falling off the edge, Globalization, World Peace and other lies". Therein are contained some frightening statistics, not just concerning the inherent national angst and violence that arise when sudden change is imposed upon a country. Take India for instance, at current levels of growth it will require more than 100 years for the nation to reach parity with living standards in the West. During the Iraq war, more journalists were killed than during the whole of the Vietnam war. While the author makes clear his distaste for the globalization agenda from the viewpoint of a hardened war correspondent, the parallels between the socio-economic-political and technological universes could not be less contrasting.

While the crux of the anti-globalization argument rests on inequality – the "haves" versus the "have not's", the resource rich versus the resource poor, the strong against the weak – technology has almost always been considered the great leveller. And with the similar ethical passion as the politician, banker or plutocrat, the argument has always been that wealth trickles down – be that financial wealth or educational knowledge and skill. It is however becoming more clear in these recent years of austerity that this model is broken – as the gap between rich and poor is widening not decreasing. And so it is with technology. Apart from the few who really want to get their hands dirty, the majority of people are happy with their Facebook, Twitter, Email and website access until of course when the communications go down or a virus attacks in which case all hell breaks loose. Our Internet connection here has been very unreliable over the past few months. Mainly I suspect this is happening due to the roll-out of Fiber in the area – at one point I was experiencing the lightening fast connection speed of 100Kbps and all without the comforting sound of a pair of modems handshaking.

My teenage daughter was climbing the walls, and life was definitely more – not less stressful.

Now, I could be accused of over dramatization here – after all, wars, rebellions and uprisings kill people – a lifeless Ipad or PC will rarely be cited as the cause of death on a coroner's certificate. But dig beneath the surface, and the same subtle dynamic is at play – you must be part of the crowd or else you are an outsider. Join the technological arms race or be flattened by the opposition. Sup at the communal bowl of the Internet of Things (IoTs). And then you may pick up some nasty diseases while you are at it. The problem is not the connectivity; it is the uncharted territory that goes with it. What is sauce for the goose is sauce for the gander and the old establishment is reeling as the openness of new media is exposing their weaknesses quicker than the traditional channels ever could. Politicians and legislators are struggling to keep hold as the criminal and terrorist move from the more visceral bank robberies and bombings to fraud and electronic infrastructure attacks. The definition of property is being redefined by mega-corporations, and personal data became public knowledge years ago. And here lies the rub – while not explicitly stated by Alex Parry, you can almost touch the intellectual exasperation on every page of his book – where has the ethical order gone? Why are we descending into chaos?

It would be easy then, to take the simplistic view and say the Internet is broken and try and regain it. This seems to be the current philosophy amongst the powers that are existing and thankfully we are not yet at the stage where we have UN peace-keepers patrolling the World Wide Web. However, the insidious creep of government vampiring our network traffic and communications is continuously unabated at the same time that Internet censorship is on the rise. As I have stated many times, politicians and lawyers are best kept as far away from regulating technology as
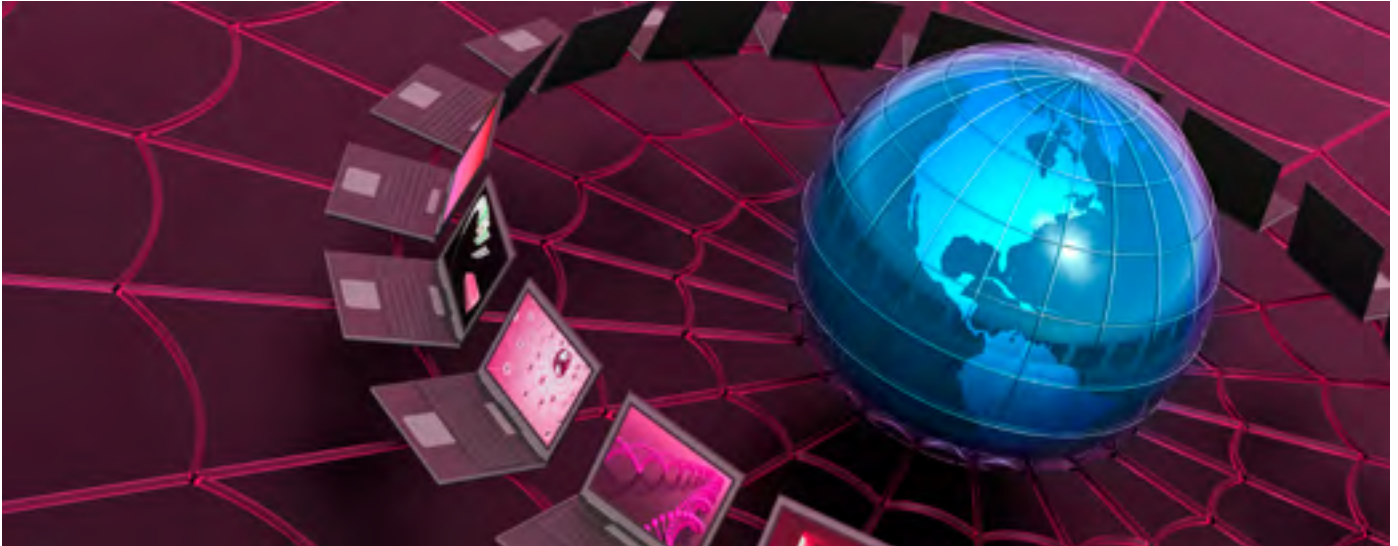
possible as they rarely appreciate the subtleties of robust engineering or human ingenuity. What is more troubling is that we can expect the same global solutions to the global problems as we have had over the past hundred years – in other words sweet nothing other than to make the problem much worse in the long term. The technological age was meant to bring in more free time, a paperless office, free electricity and a better quality of life for everyone. For a few, this is the case, but the majority of the world remains poor, hungry and dispossessed. We can send a man to the moon but can't manage to get clean drinking water to the 780 million people who need it. And where do the global leaders house their outsourcing operations? Where it is cheap – like India of course.

In the late 80's and early 90's, there was a great trend for multinationals to be ethically led. I had the pleasure of working with two visionaries – *Michael Kidron* and *Bela Hatvany* – in a small tech start-up that monitored the ethical behavior of multinationals and blue chips. Sadly, it was ahead of its time and didn't make the grade in the harsh world of pre-internet electronic publishing. Sites like *Wikileaks* today have global reach, but the pace of change is too slow and in the arms race a small organization has no traction with a huge PR machine – especially if it has access to other forms of media or worse to be the ear of a government. That is why it is so critical that the Internet remains a pure voice for all. This curse of "managing expectations" has penetrated through corporate website forums, blogs and message feeds to the extent that PR companies are now advertising on commercial radio offering to manage your reputation on the web. Software tools are used to monitor Twitter and Facebook in the guise of customer service, but you can bet that the real motive behind this is to silence any real criticism. Na-

ture abhors a vacuum, and as we have allowed commercial interests to dominate the infrastructure of the Internet, now so shall they dominate the ethical landscape as well? And as it is with the global erosion of sovereignty, so will it be with our meta-data? The majority will go along with the agenda, not realizing that the first call of any reputable recruiter or law enforcement officer will get their Facebook pages. It is very easy to click in haste and repent at leisure.

And far away from the suggestion that we need an Internet policeman, we do however need stronger ethics and vision for the Internet. Not just on the corporate level, but for the *70%* who are not online and cannot speak for themselves. From where I stand, that is the only positive point I can see – a bunch of connected, networked individuals who make a stand and say with one voice "This is not right". If the Internet looses its edginess, and becomes just another media-barons paradise on a global scale, we really will have lost any voice as the sticky ocean of mediocrity, alternative agenda, doublespeak and compliance smothers any dissenting opinion. *Mike Kidron* and *Bela Hatvany* had it right 30 years ago – *the deeds of large corporates, governments and institutions need to be truly independent-monitoring – and they need to be held to account.*

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid-eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

Siju George

Please introduce yourself to our readers.

I am a software and network engineer. Now I'm working with the National Center of Research (ICAR) area networking in Cosenza as the network manager. I graduated from the University of Calabria, specifically at the DIMES with the "Culture Lab" (*http://culture.deis.unical.it*) in the Department of Telematics. Currently, I'm collaborating with this department on several projects related to mesh networks, and with computer science associations "Hacklab Cosenza" (*http://hacklab.cosenzainrete.it/*) and "Verde Binario" (*http://www.verdebinario.org/*). Finally, I'm a freelance columnist for Italian magazines "Linux&C" (*http://www.oltrelinux.com/*) and "Linux Magazine" (*http://www.linux-magazine.it/*). I am very interested in the developments in mesh technology, security in wireless networks and the integration of complex services in MANs. I also work on systems with a focus on RTP and Asterisk VOIP made services. Now I live in Calabria, in the deep south of Italy, but I have traveled throughout my country for work for five years. Now I'm back in my hometown. I'll see what the future holds for me…

Could you tell us more about your background?

My first experience with a computer was an Atari XE System for a course in BASIC programming I took in middle school. It was the era of the legendary Commodore-64.

In high school, I worked purely on Windows machines (then 3.1 and 3.11 and MS-DOS). In 1997, I had my first encounter with Unix systems, with RED HAT 1.0, the legendary BAT. I had many difficulties installing with dual boot :).

In college, I continued to work more frequently with Linux systems until in 2003, I worked with FreeBSD for the first time. Since then, being directed by my studies to the security and performance of networks (wireless and not), I have almost completely abandoned the systems of the house of Redmond.

I have used various programming languages, such as PHP, C, C++, Java, Perl, Python, but today if it's possible, I always use shell scripting and Python.

Please tell us about your proudest achievements.

I'm happy to be involved in various activities related to the spread of Open Source software. First of all, I work as a teaching assistant in the courses of Telematics at the University of Calabria, where I like to make simple real architecture, so as to see that all the theory being studied is used to implement concrete infrastructure.

I like to work with different magazines because it allows me to extend the range of people that I can reach, as if I were in a distance course.

From the business point of view, however, I'm pleased to have completed a 20-domain migration of email services,

DNS, and web sites with hundreds of users over the course of one week for a customer. Finally for another company, I created a series of useful networks to connect thirty locations throughout the country. Obviously, I am very happy at my current job as a network administrator at the CNR.

Please tell our readers, what does the future of *BSD OSes look like?

From the first encounter, I've always admired a lot of the BSD operating systems. OpenBSD is a constant reference for anyone involved in security. I use it to implement my firewall, as well as Linux with iptables. FreeBSD is a milestone to achieve a robust and secure server, much easier to manage than older releases. NetBSD can now compete with Debian as far as the number of platforms supported via installation. So much for the 'main' distributions. Distributions such as DesktopBSD and PC-BSD are geared to novice users, with superb results in my opinion, and in addition to the stability of BSD have simple and friendly interfaces. M0nowall and pfSense are ideal for building robust and reliable firewall systems, but at the same time easy to handle, and finally FreeNAS is an excellent alternative to proprietary storage systems. From the above, I believe that BSD systems will only grow over time and earn more and more consistent slices of the market, both among individual users and big companies.

From your point of view, what are the best capabilities of *BSD?

Certainly not to underestimate the fine documentation always available online, which represents almost a peculiarity of BSD systems, some of which were among the first to get on the field a full, well-organized documentation. The system is "reasoned" and well built, with a clear division of areas "root space" and "user space". It should also be taken into great consideration the excellent system software installation, split between packages and ports, something that no other Unix distro can boast. Finally, it's necessary to recognize the reliability of BSD systems. Subject to proper management of security patches and updates, there are servers that have held up for years without significant management intervention.

What was/is your best tool to work with?

In truth, I use several tools in my daily life, such as the Nano text editor (although I often use Kate on KDE, my favorite desktop environment), GNS3 to simulate complex networks and test the operation before moving to the production environment. Those are my regular working tools. However, I also use virtualizers like VirtualBox and

KVM, as well as VMware. I tend to virtualize my servers if there are no particular contraindications, in order to manage crashes and backups better, for example. If I have to teach or write technical or research papers, then Emacs and LaTeX become my allies.

What is the best advice for those who want to use *BSD OS and why should they?

When you use a BSD-like system, it is necessary to become familiar and have a greater awareness of the system as users become experts and not "mere users" of personal computers.

BSD systems, as well as those like Linux, force users to make this initial effort, but it pays off over time with a deeper knowledge of the subject and the ability to manage systems robustly and reliably. Thanks to this, in the event you become professionals, you can manage high-level architectures. If you add to this the fact that the systems are open source and have splendid documentation, it is just time to start the download for the first installation :).

What are you looking for in terms of career development?

I would love to work in between the corporate world and the university, collaborating on the development of innovative projects while simultaneously teaching and writing publications in trade magazines.

It's complex and very challenging, I admit, but it is important to believe in what we do and what we like to do. It's advice that, in my small way, I can give to all readers.

How do you want to improve yourself in the next year?

I think I will continue my normal life, really, maybe consolidating my working relationships at the CNR. I like to look up a lot of docs online and study a lot of books if the argument proves to be interesting. In particular, I'm working on Mesh systems, and I believe that this technology has great potential. One very obvious use is to provide Internet connections in locations not covered by the national network. It can also be used to interconnect networks of sensors physically far apart for very responsive monitoring.

BY BSD TEAM