BSD MAGAZINE
FOR NOVICE AND ADVANCED USERS

OpenBSD 5.4
CONFIGURE OPENBSD BASIC SERVICES

GETTING TO GRIPS WITH THE GIMP

USER, GROUP AND PASSWORD MANAGEMENT ON LINUX AND SOLARIS

HOW SECURE CAN SECURE SHELL (SSH) BE?

SECURING CENTOS AND SOLARIS 11 WITH PUPPET

VOL.8 NO.02
ISSUE 02/2014(55)
1898-9144

systems
855-GREP-4-IX
www.iXsystems.com
Enterprise Servers and Storage for Open Source

✓ Rock-Solid Performance
✓ Professional In-House Support
High Performance, High Density Servers for Data Center, Virtualization, & HPC

http://www.iXsystems.com/e5

KEY FEATURES

<table>
<thead>
<tr>
<th>Model</th>
<th>iXR-22X4IB</th>
<th>iXR-1204 +10G</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Dual Intel® Xeon® Processors E5-2600 Family per node</td>
<td>- Dual Intel® Xeon® Processors E5-2600 Family</td>
</tr>
<tr>
<td></td>
<td>- Intel® C600 series chipset</td>
<td>- Intel® C600 series chipset</td>
</tr>
<tr>
<td></td>
<td>- Four server nodes in 2U of rack space</td>
<td>- Intel® XS40 Dual-Port 10 Gigabit Ethernet Controllers</td>
</tr>
<tr>
<td></td>
<td>- Up to 256GB main memory per server node</td>
<td>- Up to 16 Cores and 32 process threads</td>
</tr>
<tr>
<td></td>
<td>- One Mellanox® ConnectX QDR 40Gbps Infiniband w/QSFP Connector per node</td>
<td>- Up to 768GB main memory</td>
</tr>
<tr>
<td></td>
<td>- 12 SAS/SATA drive bays, 3 per node</td>
<td>- Four SAS/SATA drive bays</td>
</tr>
<tr>
<td></td>
<td>- Hardware RAID via LSI2108 controller</td>
<td>- Onboard SATA RAID 0, 1, 5, and 10</td>
</tr>
<tr>
<td></td>
<td>- Shared 1620W redundant high-efficiency Platinum level (91%+) power supplies</td>
<td>- 700W high-efficiency redundant power supply with FC and PMBus (80%+ Gold Certified)</td>
</tr>
</tbody>
</table>

Call iXsystems toll free or visit our website today! 1-855-GREP-4-IX | www.iXsystems.com
High-Density iXsystems Servers powered by the Intel® Xeon® Processor E5-2600 Family and Intel® C600 series chipset can pack up to 768GB of RAM into 1U of rack space or up to 8 processors - with up to 128 threads - in 2U.

On-board 10 Gigabit Ethernet and Infiniband for Greater Throughput in less Rack Space.

Servers from iXsystems based on the Intel® Xeon® Processor E5-2600 Family feature high-throughput connections on the motherboard, saving critical expansion space. The Intel® C600 Series chipset supports up to 384GB of RAM per processor, allowing performance in a single server to reach new heights. This ensures that you’re not paying for more than you need to achieve the performance you want.

The iXR-1204 +10G features dual onboard 10GigE + dual onboard 1GigE network controllers, up to 768GB of RAM and dual Intel® Xeon® Processors E5-2600 Family, freeing up critical expansion card space for application-specific hardware. The uncompromised performance and flexibility of the iXR-1204 +10G makes it suitable for clustering, high-traffic web servers, virtualization, and cloud computing applications - anywhere you need the most resources available.

For even greater performance density, the iXR-22X4IB squeezes four server nodes into two units of rack space, each with dual Intel® Xeon® Processors E5-2600 Family, up to 256GB of RAM, and an on-board Mellanox® ConnectX QDR 40Gbps Infiniband w/QSFP Connector. The iXR-22X4IB is perfect for high-powered computing, virtualization, or business intelligence applications that require the computing power of the Intel® Xeon® Processor E5-2600 Family and the high throughput of Infiniband.
Dear BSD Readers,

We are pleased to present you with the newest issue of BSD Magazine. In the February issue, we have decided to focus on various important aspects for Unix users.

Inside, you will find interesting articles, such as Configure OpenBSD 5.4 Basic Services. In short, thanks to reading this article, you will learn what you need to do to configure a vether (Virtual Ethernet Device Driver) to be able to provide NAT for your PPTP clients.

The next article in the newest issue is entitled Getting to Grips with the Gimp. In Rob’s new series on image manipulation and design, you will look at graphic design basics, and you will learn how to use the most popular Open Source graphics software – The Gimp.

In the following section, that is Unix, you will find an article entitled User, Group and Password Management on Linux and Solaris and Securing CentOS and Solaris 11 with Puppet. The first one will cover the user, group and password management tools and the second one will provide you with detail about how security can be managed on CentOS 6.x and Solaris 11.1 hosts with Puppet 3.x.

We would also like to encourage you to read the interview with Peter N.M. Hansteen in the February Issue.

Thank you BSD fans for your invaluable support and contribution.

Enjoy reading!

BSD Team
### OpenBSD 5.4

**06 Configure OpenBSD 5.4 Basic Services**  
*Wesley Mouedine Assaby*

The webserver has only one nic, so we need to configure a vether (Virtual ethernet device driver) to be able to provide NAT for our PPTP clients. It is connected to the Internet through a simple modem-router. We use OpenBSD 5.4. Tested with Apple, Samsung phones, and a laptop running Windows 8: PPTP connection / reach a webpage hosted by the webserver.

### Security

**10 How Secure can Secure Shell (SSH) be?**  
*Arkadiusz Majewski*

SSH, and especially OpenSSH, are very powerful applications when beginners use a Unix-like or Linux operating system. It is very useful for administrators to secure access to the system and improve scalability to whole networks. I hope this article on OpenSSH expanded your knowledge and challenges you to use it. Try to employ it in your next project. The article concentrates on SFTP (SSH File Transfer Protocol) supported by OpenSSH and sftp-server subsystem, but has useful information for a standard file transfer preferring SFTP to FTP (File Transfer Protocol).

### UNIX

**22 User, Group and Password Management on Linux and Solaris**  
*Toki Winter*

This article will cover the user, group and password management tools available on the Linux and Solaris Operating Systems. The specific versions covered here are CentOS 6.4 and Solaris 11.1, though the commands will transfer to many other distributions without modifications (especially RHEL and its clones), or with slight alterations to command options. Check your system documentation and manual pages for further information.

**30 Securing CentOS and Solaris 11 with Puppet**  
*Toki Winter*

Puppet is system administration automation software from Puppet Labs (http://puppetlabs.com). It has gained a lot of popularity, and rivals other automation/orchestration software such as Chef and Ansible. In this article, Toki will detail how security can be managed on CentOS 6.x and Solaris 11.1 hosts with Puppet 3.x. Some familiarity with Puppet or some other automation software, as well as a Linux/UNIX system administrator audience, is assumed.

### GIMP

**12 Getting to Grips with the Gimp – Part 1**  
*Rob Somerville*

It might seem strange having a “non-technical” how-to series, but in this age of digital photography, graphics intensive website design and visual icons, more and more emphasis is being placed on imagery as a method of communication. Good graphic design is also useful for presentations, flyers, and publications; the list is endless. Some people just lift images from Google or make use of professional stock images, the latter being expensive and the former dubious from a copyright perspective. What can be more satisfying than manipulating and creating your own artwork? In our new series on image manipulation and design, we will look at graphic design basics, and how to use the most popular Open Source graphics software – The Gimp.

### Interview

**50 Interview with Peter N. M. Hansteen**  
*BSD Team*

Peter N. M. Hansteen is a consultant, writer and sysadmin from Bergen, Norway. A longtime freenix advocate and during recent years a frequent lecturer and tutor with emphasis on FreeBSD and OpenBSD, author of several articles and “The Book of PF”. He writes a frequently slashdotted blog at http://bsdly.blogspot.com/.

### Column

**56 With the collapse of Red Flag Software (the world’s second-largest Linux distributor) is the dream of Linux on the Desktop even further out of reach?**  
*Rob Somerville*
Configure OpenBSD 5.4

Basic Services

The webserver has only one nic, so we need to configure a vether (Virtual ethernet device driver) to be able to provide NAT for our PPTP clients. It is connected to the internet through a simple modem-router. We use OpenBSD 5.4. Tested with Apple, Samsung phones, and a laptop running Windows 8: PPTP connection / reach a webpage hosted by the webserver.

What you will learn…

• Configure OpenBSD basic services.
• Understand Packet Filter.
• Build a PPTP vpn server.
• How to use vether

What you should know…

• Basic TCP/IP knowledge, OpenBSD installation and post-configuration.

The role-play: The webserver has only one nic, so we need to configure vether to be able to provide NAT for our PPTP clients. It is connected to the Internet through a simple modem-router. We use OpenBSD 5.4-RELEASE-i386 on the webserver. Tested with Apple, Samsung phones, and a laptop running Windows 8: PPTP connection / reach a webpage hosted by the webserver. First, read the man pages for, PF.CONF(5), PFCTL(8), NPPPDEV(8), NPPPCTL(8), PPPX(4), PIPEX(4), GRE(4), VETHER(4). Make sure the webserver is connected to the Internet.

Connect to webserver through a PPTP VPN

OpenBSD 5.4 webserver
PPTP with 1 nic: 192.168.218.54

modem-router 192.168.218.1/24

Use a webserver through a vpn pppip
Update with a fresh copy of OpenBSD and install to -stable, using Openup

# Get it
ftp https://stable.mtier.org/openup

# Run it
./openup

# You need to reboot if the kernel has been replaced

Set the kernel state (reboot is not needed)

# Permit forwarding (routing) of IPv4 packets
sysctl net.inet.ip.forwarding=1

# Allow GRE packets in and out of the system
sysctl net.inet.gre.allow=1

# Enable pipex (used with tun and pppx)
sysctl net.pipex.enable=1

# Do not forget to enable them in the file /etc/sysctl.
  conf to keep these settings at reboot.

Configure vether with this address :
172.17.2.54/24 (it is my choice)

# Create interface vether0
echo "inet 172.17.2.54 255.255.255.0" > /etc/hostname.
  vether0
sh /etc/netstart vether0

# Verify
ifconfig vether0

# By default, vether0 is associated to a group named
  vether
# And the internet interface is associated to the egress
  group

Configure Packet-Filter (/etc/pf.conf)

# No filters on loopback interface
set skip on lo

# We do not want to load fingerprints
set fingerprints "/dev/null"

# NAT for PPTP clients
match out on egress inet from vether:network to any
  nat-to egress

# Policy : block all and log all blocked packets
block log all
# We trust outbound
pass out

# PPTP traffic
pass in on vether
pass proto gre
pass on pppx0
pass in on egress inet proto tcp from any to any
   PPTP

# Permit computers in our local network to use our webserver
pass in on egress inet proto tcp from any to any
   www

Load the new ruleset
/sbin/pfctl -vf /etc/pf.conf

Configure npppd authentication using the file /etc/npppd/npppd-users, this last one contains:
   # a username wesley and his password welCom3
   wesley:
      :password=welCom3:

Configure npppd authentication using the file /etc/npppd/npppd-users, this last one contains:
   # a username wesley and his password welCom3
   wesley:
      :password=welCom3:

Configure npppd (/etc/npppd/npppd.conf)
   authentication LOCAL type local {
      users-file "/etc/npppd/npppd-users"
   }

tunnel VPN protocol PPTP {
   listen on 0.0.0.0
}

ipcp IPCP {
   pool-address 172.17.2.100-172.17.2.150
   dns-servers 8.8.8.8
}

interface pppx0 address 172.17.2.1 ipcp IPCP
   Bind tunnel from VPN authenticated by LOCAL to pppx0

---

Configure npppd authentication using the file /etc/npppd/npppd-users, this last one contains:
   # a username wesley and his password welCom3

   wesley:
      :password=welCom3:

Start npppd
   echo "npppd_flags=" >> /etc/rc.conf.local
   /etc/rc.d/npppd start

   # For troubleshootings : tail -f /var/log/daemon &
   /etc/rc.d/npppd start

# Verify that it listens on port 1723 (PPTP)
   netstat -anf inet | grep 1723

   Do not forget to open the port 1723 TCP in the modem-router (Port forwarding from Any to 192.168.218.54:1723 TCP).

Start apache (webserver)
   echo "httpd_flags=" >> /etc/rc.conf.local
   /etc/rc.d/httpd start

   # Try on a computer in the local network
   http://192.168.218.54
   # On PPTP clients : http://172.17.2.54

   To connect a client, use the following information:
   PPTP connection / IP: aa.bb.cc.dd / Username: wesley / and password: welCom3

   view connected clients (on the webserver)
   npppctl session all

---

Conclusions
The trick is to use vether(4), and now we can provide nat for our PPTP clients.

---

WESLEY MOUEDINE ASSABY

Wesley MOUEDINE ASSABY lives in Reunion island, near Mauritius. He works as a network administrator at AISE-INFORMATIQUE (http://www.aise.re) where he installs some firewalls (Soekris appliances) and mail servers, all using OpenBSD systems. He has used OpenBSD since 2007.

To contact the author write at wesley [at] mouedine [dot] net

© Copyright 2013
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

www.titania.com
First, and most important, is security. SFTP is supported by OpenSSH and is secure by default. Traffic between client and server is encrypted.

How to configure WinSCP client for SFTP was mentioned in the first article of the series named: Basic Configuration of OpenSSH (issue 11/2013). You can use private/public keys and one time passwords just like a normal SSH connection. Be informed that the application Locker (www.iptrace.pl and Download->Locker) does not support WinSCP client and others, so if you are going to use your server as an SFTP server you need to disable and comment the locker application in the .profile file.

When you configure SSH, SFTP is enabled by default and can be used simultaneously with SSH terminal access, VPNs, etc. The system service responsible for SFTP is a subsystem sftp-server non-standalone system, but is ready to work with the sshd daemon and has its own configuration in the sshd_config file. The main option in this file is:

```
subsystem sftp /usr/libexec/sftp-server
```

If you comment out the above line, you can still use your system as an SFTP server.

The rest of the sshd_config file can be the same for SSH terminal connections. If you want to apply the file transfer for more than one or two users, just add the users to the appropriate option in the sshd_config file.

Sometimes it is required to use SFTP in read-only mode. In this case use the uncommented line above. Add the sftp-server option –R to deny the writing of any data for every user. The option with values is shown below:

```
subsystem sftp /usr/libexec/sftp-server -R
```

Some users may have read access to one or more directories and can copy some files, especially configuration files, and have the ability to read and find bugs in your configurations that are only useful for internal administrators.
For the rest of the users, separate directories can be used for each user, or group, to log in. One time passwords must be disabled, because they cannot be used in this case. How to disable OTP was mentioned in the second article of this series named: One Time Password aka OTP (issue 12/2013).

Add the following lines to your sshd_config file. You can specify group or user for such access.

```plaintext
Subsystem sftp internal-sftp

Match Group sftpusers
    ChrootDirectory %h
    ForceCommand internal-sftp
    AllowTcpForwarding no

Match User username
    ChrootDirectory %h
    ForceCommand internal-sftp
```

Sometimes it is desirable to allow a group to have read-only access to files for a particular user. In this case you can just use standard chown and chmod commands.

**Conclusion**

SSH, and especially OpenSSH, are very powerful applications when beginners use a Unix-like or Linux operating system. It is very useful for administrators to secure access to the system and improve scalability to whole networks. I hope this article on OpenSSH expanded your knowledge and challenges you to use it. Try to employ it in your next project.

**ARKADIUSZ MAJEWSKI, BENG**

Arkadiusz Majewski comes from Poland. He has 15 years' experience with ICT technologies, including 15 years of IT networks, 10 years of BSD systems and MS Windows Server solutions. He has also 5 years' experience with programming languages and Telco solutions. He's interested in security on all business and ICT levels. In his free time he reads ICT books and deepens his knowledge about science (math, physics, chemistry). His hobbies are cycling and motorization. He's a graduate of Warsaw Information Technology under the auspices of the Polish Academy of Sciences. He's the IT Manager at an international company. Feel free to contact the author via e-mail at bsd.magazine@iptrace.pl.
Getting to Grips with the Gimp – Part 1

In our new series on image manipulation and design, we will look at graphic design basics, and how to use the most popular Open Source graphics software – The Gimp.

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

What you should know…
• General PC administration skills

It might seem strange having a “non-technical” how-to series, but in this age of digital photography, graphics intensive website design and visual icons, more and more emphasis is being placed on imagery as a method of communication. Good graphic design is also useful for presentations, flyers, and publications; the list is endless. Some people just lift images from Google or make use of professional stock images, the latter being expensive and the former dubious from a copyright perspective. What can be more satisfying than manipulating and creating your own artwork?

I first became hooked on graphics programs in the mid-eighties when I got my hands on an Amiga and Deluxe Paint. Sadly no more, I spent years working with other vector based programs such as Corel Draw, Arts and Letters, etc. until I came across the Gimp in the early days of Open Source. While Adobe Photoshop has always been around, it was (and still is) prohibitively expensive for the

Figure 1. Raster and Vector images
amateur design enthusiast, and until the Gimp arrived there was no real raster based alternative.

The Gimp (the GNU Image Manipulation Program) can be used for photo retouching, image authoring and a host of other functions including creating animated GIFs, etc.

While both vector and raster based programs have their uses, the former is mainly used for posters, logos and artwork that requires high definition at high resolutions. Gimp on the other hand works at the pixel level and therefore is suitable for image manipulation. For vector graphics manipulation, Inkscape is an excellent Open Source tool. [See Figure 1 – Raster and Vector graphics].

Requirements
The Gimp is available for Mac (OSX), Windows and virtually every flavour of Linux and *BSD. While version numbers may vary slightly across platforms, I will be using 2.8.4 for this tutorial though some platforms may still be on 2.6.x. There are some subtle differences between the two versions (e.g. window docking, file import and export, etc.) but the majority of functions are the same. What is more important than the version is the PC you run the software on. You will need plenty of RAM if you are going to be working with large images. A good quality graphics card and monitor are also important, but most modern kits will be fine. The biggest issue is colour drift and lighting – editing an image on a CRT monitor under fluorescent light will be a different visual experience from using an LCD or LED monitor under tungsten lighting. One of the reasons why graphic designers are fanatical about Mac’s is the excellent colour balance and font support, something that is not consistent across different manufacturers.

Figure 2. Default Gimp layout

Figure 3. Single window mode
Also, it is important to respect copyright and attribute credit where it is due. All images used in this series will either come from the author’s own collection, royalty free from http://www.sxc.hu or under a Creative Commons licence.

Your Chance to Contribute
If you have an image you would like manipulated, or have some ideas for the series, please contact me via BSD magazine. While my favourite task is taking mundane images and applying liberal doses of satire, surrealism or atmosphere, I am open to suggestions and any commissions from readers.

Let’s get started
Install Gimp on a PC and platform of your choice, either via your package management system or by download from www.gimp.org/downloads. Upon opening the Gimp, you will be presented with multiple windows [Figure 2]. As I am left handed and I don’t like multiple floating windows cluttering my desktop, I have selected Windows → Single-Window mode [Figure 3]. I have also moved all the tabs from Layers, Brushes, Gradients etc. across to the left hand side dock, and expanded the width slightly so all the controls are visible [Figure 4]. You may want to tweak the default settings as well [Figure 5 – 8].
Getting to Grips with the Gimp – Part 1

A list of the major tools and functions is listed in Table 1. In the belated spirit of St Valentines day, we will modify a picture of a rose and add a shadow using a mask and multiple layers to produce the resulting Image 2. These two tools are very powerful, and quickly allow the designer to transform an image with ease.

**Figure 7.** Set the default template size

**Figure 8.** Increase undo levels and undo memory

**Editing an image**

A list of the major tools and functions is listed in Table 1. In the belated spirit of St Valentines day, we will modify a picture of a rose and add a shadow using a mask and multiple layers to produce the resulting Image 2. These two tools are very powerful, and quickly allow the designer to transform an image with ease.

**Image 1.** rose-with-bud-ii-1436558-m.jpg

**Image 2.** The final picture
Step 1
Download rose-with-bud-ii-1436558-m.jpg (Image 1) from the website listed in Table 2. Open in the Gimp using File → Open

Step 2
Rotate the image with Image → Transform → Rotate 90 Degrees anti-clockwise. Zoom in by pressing + a couple of times or click on the image with the Zoom tool [Figure 9].

Step 3
Using the fuzzy select tool, click on the image at position 100px x 50px to make a selection. You will see a boundary of “marching worms” [Figure 10].

Step 4
Click on the Toggle quick Mask icon just to the bottom Left Hand Side of the image or press Shift Q. A red mask will cover the areas that will not be affected by our changes [Figure 10].
Step 5
Using the erase tool, and increasing / decreasing the size of the brush as required, remove the mask from the background to leave the rose, the stem and a few leaves. Zoom in and out as required (+ / –), and don’t worry if you overshoot slightly. Either press Ctrl Z to undo, or retouch with the paintbrush tool. I used the 2. Hardness 025 brush circular, but choose a brush you feel comfortable with. The final result should look like Figure 11. [Figure 11 – 12].
Step 6
Un-toggle the Quick Mask and the flower with leaves should be selected [Figure 13].

Step 7
Press Ctrl, and the background will be filled with a black background. Press Shift Ctrl A to deselect the background and zoom out to 100% [Figure 14].

Figure 13. Selected area with “marching worms”

Figure 14. Rose on black background
Step 8
From the menu Image → Canvas size resize the image to 300px x 410px. Ensure the chain between width and height is broken and that Resize layers → All layers is selected. Click on the layers icon and add a new black layer. Drag this new layer down so the rose is the top layer. Right click on the rose layer and add transparency by adding a new Alpha channel, then using the rectangular select tool, highlight the white area beneath the rose. Press Del to delete this selection, then press Ctrl I to select the upper part of the image. Press Ctrl C to copy, click on the lower black layer and press Ctrl V to paste the selection. Using the move tool, adjust the copy of the rose so that it is roughly below the first top rose, then select Layer → Transform → Flip vertically. Use the move tool to adjust the position of the lower layer so that approximately 1/3rd of the inverted rose is showing. When satisfied, right click on the floating selection and choose Anchor layer. [Figure 15].

Figure 15. Rose with inverted reflection before top layer erased and transparency / blur applied

Step 9
Click on the top rose layer. Using the erase tool, remove just enough of the top black area to make the reflection look convincing. Add a new black layer with 65% transparency between the rose and the reflection. Select the bottom layer and blur by 6px using Filters → Blur → Gaussian blur. Crop the image using the crop tool and the finished result can be seen in Image 2.
<table>
<thead>
<tr>
<th>Tool</th>
<th>Description and usage</th>
<th>Tool</th>
<th>Description and usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rectangle select.</td>
<td></td>
<td>Scale</td>
<td>Perspective clone</td>
</tr>
<tr>
<td>Ellipse select</td>
<td></td>
<td>Shear</td>
<td>Blur / sharpen</td>
</tr>
<tr>
<td>Free select</td>
<td></td>
<td>Perspective</td>
<td>Smudge</td>
</tr>
<tr>
<td>Fuzzy select</td>
<td></td>
<td>Flip</td>
<td>Dodge / burn</td>
</tr>
<tr>
<td>Colour select</td>
<td></td>
<td>Cage transform</td>
<td>Foreground / background colours</td>
</tr>
<tr>
<td>Scissors select</td>
<td></td>
<td>Text</td>
<td>Layers</td>
</tr>
<tr>
<td>Foreground select</td>
<td></td>
<td>Bucket fill</td>
<td>Tool options</td>
</tr>
<tr>
<td>Paths tool</td>
<td></td>
<td>Blend</td>
<td>Brushes</td>
</tr>
<tr>
<td>Colour picker</td>
<td></td>
<td>Pencil</td>
<td>Patterns</td>
</tr>
<tr>
<td>Zoom</td>
<td></td>
<td>Paintbrush</td>
<td>Gradients</td>
</tr>
<tr>
<td>Measure</td>
<td></td>
<td>Eraser</td>
<td>Channels</td>
</tr>
<tr>
<td>Move</td>
<td></td>
<td>Airbrush</td>
<td>Paths</td>
</tr>
<tr>
<td>Alignment</td>
<td></td>
<td>Ink</td>
<td>Undo history</td>
</tr>
<tr>
<td>Crop</td>
<td></td>
<td>Clone</td>
<td>Configure tab</td>
</tr>
<tr>
<td>Rotate</td>
<td></td>
<td>Healing</td>
<td></td>
</tr>
</tbody>
</table>
Table 2. Details and credits

<table>
<thead>
<tr>
<th>Image</th>
<th>URL</th>
<th>Details and credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image 1</td>
<td><a href="http://www.sxc.hu/photo/1436558">http://www.sxc.hu/photo/1436558</a></td>
<td>ROSE with BUD II Rose blossom and nearby bud in vibrant color hues. Uploaded by lance1</td>
</tr>
</tbody>
</table>

Resources

- Search Creative commons – http://search.creativecommons.org
- Deviant art – http://www.deviantart.com
- Stock.xchng – http://www.sxc.hu

In the next article
We will look at improving our reflected image and lighting, shade and dark.

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.

Better safe than sorry
www.demyo.com
Securing CentOS and Solaris 11 with Puppet

Puppet is system administration automation software from Puppet Labs (http://puppetlabs.com). It has gained a lot of popularity, and rivals other automation/orchestration software such as Chef and Ansible.

What you will learn...
- How security can be managed on CentOS and Solaris with Puppet.
- How to configure Puppet.
- How to deploy a set of security configurations to the hosts.

What you should know...
- Basic security knowledge.

In this article, I will detail how security can be managed on CentOS 6.x and Solaris 11.1 hosts with Puppet 3.x. Some familiarity with Puppet or some other automation software, as well as a Linux/UNIX system administrator audience, is assumed.

The topology being used for the examples given in this article is shown in Figure 1.

As you can see, centosa is the Puppet master. Four hosts will contact it for configuration, including itself. There are three CentOS hosts in total (centos[a-c]) and a single Solaris host (sol11test). We will start with server and agent installation, then move on to cover various Puppet configuration tasks, and develop our own security module to deploy a set of security configurations to the hosts.

Server Installation
The Puppet server is installed on host centosa. Start by installing the latest repository RPM from http://docs.puppetlabs.com/guides/puppetlabs_package_repositories.html#for-red-hat-enterprise-linux-and-derivatives. At the time of writing, this was puppetlabs-release-6-7.noarch.rpm.

```
# rpm -ivh https://yum.puppetlabs.com/el/6/products/x86_64/puppetlabs-release-6-7.noarch.rpm
```

Whilst this article has been written with CentOS 6.x and Solaris 11.1 in mind, the techniques utilised should translate to RHEL/OEL 6.x and Solaris 10 without many changes. In case of doubt, consult the relevant security guide for your operating system at http://cisecurity.org.

Figure 1. Example Puppet topology
Let’s see what was just installed:

```
# rpm -ql puppetlabs-release
/etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs
/etc/yum.repos.d/puppetlabs.repo
```

The appropriate repositories are enabled by default (check `/etc/yum.repos.d/puppetlabs.repo` for details):

```
[puppetlabs-products]
name=Puppet Labs Products El 6 – $basearch
baseurl=http://yum.puppetlabs.com/el/6/products/$basearch
gpgkey-file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs
enabled=1
gpgcheck=1
[puppetlabs-deps]
name=Puppet Labs Dependencies El 6 – $basearch
baseurl=http://yum.puppetlabs.com/el/6/dependencies/$basearch
gpgkey-file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs
enabled=1
gpgcheck=1
```

Next, install the Puppet server and agent packages, and their dependencies. This will install various required packages such as `ruby`, `facter`, `hiera`, and others.

```
# yum -y install puppet-server puppet
```

The packages installed on a minimal CentOS installation are as shown in Listing 1.

Once the packages are installed, the Puppet master can be started. We will use the init scripts supplied with the Puppet master to control the daemon, and `chkconfig` to have it run at the appropriate runlevels.

First, start the Puppet master service:

```
# service puppetmaster start
Starting puppetmaster:                         [  OK  ]
```

Next, use `chkconfig` to enable the service:

```
# chkconfig puppetmaster on
```

Confirm that the service is configured to start as intended:

---

### Listing 1. The Packages

<table>
<thead>
<tr>
<th>Package</th>
<th>Arch</th>
<th>Version</th>
<th>Repository</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>puppet</td>
<td>noarch</td>
<td>3.3.2-1.el6</td>
<td>puppetlabs-products</td>
<td>1.1 M</td>
</tr>
<tr>
<td>puppet-server</td>
<td>noarch</td>
<td>3.3.2-1.el6</td>
<td>puppetlabs-products</td>
<td>23 k</td>
</tr>
<tr>
<td>auges-libs</td>
<td>x86_64</td>
<td>1.0.0-5.el6</td>
<td>base</td>
<td>308 k</td>
</tr>
<tr>
<td>compat-readline5</td>
<td>x86_64</td>
<td>5.2-17.1.el6</td>
<td>base</td>
<td>130 k</td>
</tr>
<tr>
<td>dmidecode</td>
<td>x86_64</td>
<td>1:2.11-2.el6</td>
<td>base</td>
<td>71 k</td>
</tr>
<tr>
<td>facter</td>
<td>x86_64</td>
<td>1:1.7.3-1.el6</td>
<td>puppetlabs-products</td>
<td>85 k</td>
</tr>
<tr>
<td>hiera</td>
<td>noarch</td>
<td>1.3.0-1.el6</td>
<td>puppetlabs-products</td>
<td>23 k</td>
</tr>
<tr>
<td>libselinux-ruby</td>
<td>x86_64</td>
<td>2.0.94-5.3.el6_4.1</td>
<td>base</td>
<td>99 k</td>
</tr>
<tr>
<td>pciutils</td>
<td>x86_64</td>
<td>3.1.10-2.el6</td>
<td>base</td>
<td>85 k</td>
</tr>
<tr>
<td>ruby</td>
<td>x86_64</td>
<td>1.8.7.352-13.el6</td>
<td>updates</td>
<td>534 k</td>
</tr>
<tr>
<td>ruby-augeas</td>
<td>x86_64</td>
<td>0.4.1-1.el6</td>
<td>puppetlabs-deps</td>
<td>21 k</td>
</tr>
<tr>
<td>ruby-irb</td>
<td>x86_64</td>
<td>1.8.7.352-13.el6</td>
<td>updates</td>
<td>314 k</td>
</tr>
<tr>
<td>ruby-libs</td>
<td>x86_64</td>
<td>1.8.7.352-13.el6</td>
<td>updates</td>
<td>1.6 M</td>
</tr>
<tr>
<td>ruby-rdoc</td>
<td>x86_64</td>
<td>1.8.7.352-13.el6</td>
<td>updates</td>
<td>377 k</td>
</tr>
<tr>
<td>ruby-rgen</td>
<td>noarch</td>
<td>0.6.5-1.el6</td>
<td>puppetlabs-deps</td>
<td>87 k</td>
</tr>
<tr>
<td>ruby-shadow</td>
<td>x86_64</td>
<td>1.4.1-13.el6</td>
<td>puppetlabs-deps</td>
<td>11 k</td>
</tr>
<tr>
<td>rubygem-json</td>
<td>x86_64</td>
<td>1.5.5-1.el6</td>
<td>puppetlabs-deps</td>
<td>763 k</td>
</tr>
<tr>
<td>rubygems</td>
<td>noarch</td>
<td>1.3.7-5.el6</td>
<td>base</td>
<td>207 k</td>
</tr>
<tr>
<td>virt-what</td>
<td>x86_64</td>
<td>1.11-1.2.el6</td>
<td>base</td>
<td>24 k</td>
</tr>
</tbody>
</table>
Now we need to update the local firewall (iptables is enabled by default on a minimal CentOS install). The default ruleset is as follows:

```bash
# iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num target     prot opt source               destination
1  ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
  state RELATED,ESTABLISHED
2  ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0
3  ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
4  ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0
  state NEW tcp dpt:22
5  REJECT     all  --  0.0.0.0/0            0.0.0.0/0
  reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT)
num target     prot opt source               destination
1  REJECT     all  --  0.0.0.0/0            0.0.0.0/0
  reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT)
num target     prot opt source               destination
```

All of my test hosts are on the 10.1.1.0/24 network, and the Puppet master listens on port 8140. I therefore insert the rule as follows:

```bash
# iptables -I INPUT 5 -m state --state NEW \
  -p tcp --dport 8140 -s 10.1.1.0/24 -j ACCEPT
And verify:

# iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num target     prot opt source               destination
1  ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
  state RELATED,ESTABLISHED
2  ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0
3  ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
4  ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0
  state NEW tcp dpt:8140
5  ACCEPT     tcp  --  10.1.1.0/24          0.0.0.0/0
  state NEW tcp dpt:8140
...
```

If the ruleset looks good, save it:

```bash
# service iptables save
iptables: Saving firewall rules to /etc/sysconfig/iptables:[OK]
```

From another host on the network, try using the `openssl s_client` to connect to the Puppet master:

```bash
# openssl s_client -connect localhost:8140 -showcerts 2>&1
<dev/null |\n>     egrep -i '(issuer|subject)'
subject=/CN=centosa.local
issuer=/CN=Puppet CA: centosa.local
```

All Puppet traffic is encrypted over SSL.

The Puppet server is now configured. We will use the common name shown above, centosa.local, in the server values in the [agent] section of puppet.conf, so take a note of yours. To avoid issues later on, this should match the FQDN of your host.

**CentOS Agent Installation**

On each server you wish to install the agent on (in our case, centos{a,b,c}.local), perform the following steps. Note that you don’t need to reinstall the packages on your Puppet master as you already did them during the Server Installation phase above.

Install the repository RPM:

```bash
# rpm -ivh https://yum.puppetlabs.com/el/6/products/
x86_64/puppetlabs-release-6-7.noarch.rpm
```

Next, install Puppet and all dependencies. There is no need to install the puppet-server package on the other client nodes.

```bash
# yum -y install puppet
```

Once complete, on all client nodes, update `/etc/puppet/puppet.conf`. Edit (or add if it doesn’t exist) the [agent] section and add the following:

```plaintext
server = centosa.local
```

Change the value of the server variable to suit your environment. This should be the FQDN of the host, and should match the CN of your SSL certificate.

Next, a client certificate needs to be generated and signed by the Puppet master to authorize the addition of each node to the orchestration topology. Issue the `puppet agent` command with the `--test` option. `--test` includes many options useful for testing, including `--debug` and `--no-daemonize` and `--show_diff`.

```bash
# puppet agent --test
Info: Creating a new SSL key for centosb.local
Info: Caching certificate for ca
Info: Creating a new SSL certificate request for centosb.local
```

```bash
# openssl s_client -connect localhost:8140 -showcerts 2>&1
<dev/null |\n>     egrep -i '(issuer|subject)'
subject=/CN=centosa.local
issuer=/CN=Puppet CA: centosa.local
```

All Puppet traffic is encrypted over SSL.
Securing CentOS and Solaris 11 with Puppet

Exiting; no certificate found and waitforcert is disabled

We didn’t use --waitforcert (another option to puppet agent), so the agent will terminate after sending its CSR to the Puppet master for signing.

On the Puppet master, sign the outstanding request:

```bash
# puppet cert sign centosb.local
```

Notice: Signed certificate request for centosb.local
Notice: Removing file Puppet::SSL::CertificateRequest centosb.local at '/var/lib/puppet/ssl/ca/requests/centosb.local.pem'

And verify:

```bash
# puppet cert list --all
```

Listing 2. Installation

```
root@sol11test:~# pkgadd -d http://get.opencsw.org/now

## Downloading...
..............25%..............50%..............75%...... 
..............100%
## Download Complete

The following packages are available:
1  CSWpkgutil     pkgutil – Installs Solaris packages easily 
   (all) 2.6.6,REV=2013.11.12

Select package(s) you wish to process (or ‘all’ to process all packages). (default: all) [?,??,q]: all

Processing package instance <CSWpkgutil> from <http://get.opencsw.org/now>

pkgutil – Installs Solaris packages easily(all) 2.6.6,REV=2013.11.12

Please see /opt/csw/share/doc/pkgutil/license for license information.

## Processing package information.
## Processing system information.
## Verifying package dependencies.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.
This package contains scripts which will be executed with super-user permission during the process of installing this package.

Do you want to continue with the installation of <CSWpkgutil> [y,n,?] y

Installing pkgutil – Installs Solaris packages easily as <CSWpkgutil>

## Processing part 1 of 1.
/etc/opt/csw/pkgutil.conf.CSW
/opt/csw/bin/pkgutil
/opt/csw <implied directory>
/opt/csw/etc/pkgutil.conf.CSW
/opt/csw/etc <implied directory>
/opt/csw/libexec/pkgutil/wget-1386
/opt/csw/libexec/pkgutil/wget-sparc
/opt/csw/share/doc/pkgutil/license
/opt/csw/share/doc/pkgutil/readme
/opt/csw/share/man/man1/pkgutil.1
/opt/csw/var/pkgutil/admin.CSW

[ verifying class <none> ]

## Executing postinstall script.
Copying sample pkgutil.conf to /opt/csw/etc.
Copying sample pkgutil.conf to /etc/opt/csw.
Copying sample admin from /opt/csw/var/pkgutil to /var/ opt/csw/pkgutil.

NOTE!
NOTE! Make sure to check out any changes in /etc/opt/ csw/pkgutil.conf.CSW.

Installation of <CSWpkgutil> was successful.
Unix

Note above that a signed certificate already exists for centosa.local. It’s also noteworthy that the certificate has alternate DNS names of puppet and puppet.local – so we could reference centosa.local by a CNAME of puppet.local, and update the server variable in puppet.conf appropriately. Repeat the process for all nodes. You should get a clean run once the certificate has been signed:

```bash
# puppet agent --test
Info: Caching certificate for centosc.local
Info: Caching certificate_revocation_list for ca
Info: Retrieving plugin
Info: Caching catalog for centosc.local
Info: Applying configuration version ‘1386191583’
Info: Creating state file /var/lib/puppet/state/state.yaml
Notice: Finished catalog run in 0.04 seconds
```

Solaris Agent Installation
The easiest way to install Puppet on Solaris is to obtain the packages from http://OpenCSW.org. OpenCSW uses a tool called pkgutil on top of the existing Solaris toolset to obtain, install and maintain OpenCSW packages. Start by installing the latest version of CSWpkgutil: Listing 2.

The first step is to configure pkgutil to use PGP cryptographic verification. Issue the following command to install the CSWpki package via pkgutil:

```bash
# pkgutil -y -i cswpki
```

Next, import the keys with cswpki:

```bash
# cswpki --import
Do you want to import the key used for: catalog signing 2011-09?
Yes/No: Yes
```

Importing the key used for: catalog signing 2011-09

```bash
gpg: keyring '/var/opt/csw/pki/secring.gpg' created
gpg: keyring '/var/opt/csw/pki/pubring.gpg' created
gpg: key 9306CC77: public key "OpenCSW catalog signing <board@opencsw.org>" imported
```

Do you want to import the key used for: legacy catalog verification?
Yes/No: Yes

```bash
# pkgutil --fingerprint board@
cswpki --import
Do you want to import the key used for: legacy catalog verification?
Yes/No: Yes
```

Importing the key used for: legacy catalog verification

```bash
gpg: key E12E9D2F: public key "Distribution Manager <dm@blastwave.org>" imported
```

The current fingerprint is available at http://www.opencsw.org/manual/for-administrators/getting-started.html, and currently looks like this:

```bash
# gpg --homedir=/var/opt/csw/pki --fingerprint board@
```

```bash
pub 1024D/9306CC77 2011-08-31
Key fingerprint = 4DCE 3C80 AAB2 CAB1 E60C 9A3C 05F4 2D66 9306 CC77
```

With the key imported, edit /etc/opt/csw/pkgutil.conf and uncomment the following values, thus setting them to true from their defaults of false:

```bash
use_gpg=true
use_md5=true
```

Now, run a pkgutil catalog update. You should see the GPG verification taking place:

```bash
# pkgutil -U
=> Fetching new catalog and descriptions (http://mirror.opencsw.org/opencsw/testing/i386/5.11) if available ...
Checking integrity of /var/opt/csw/pkgutil/catalog.mirror.opencsw.org_opencsw_testing_i386_5.11 with gpg.
```

```bash
gpg: Signature made Wed Dec 18 10:43:20 2013 EST using DSA key ID 9306CC77
```

```bash
gpg: Good signature from "OpenCSW catalog signing <board@opencsw.org>"
```

```bash
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: 4DCE 3C80 AAB2 CAB1 E60C 9A3C 05F4 2D66 9306 CC77
```

```bash
=> 3807 packages loaded from /var/opt/csw/pkgutil/catalog.mirror.opencsw.org_opencsw_testing_i386_5.11
```

Now, we can search for the appropriate Puppet package using pkgutil -a:
As this is only a client, we will need the `puppet3` package, and any dependencies. `pkgutil` takes care of dependency resolution for us with respect to other OpenCSW.org packages. For the sake of convenience, at this point you should update your `PATH` accordingly to find binaries under `/opt/csw/bin`:

```bash
# vi ~/.profile
...
export PATH=$PATH:/opt/csw/bin
...
# . ~/.profile
```

Install the `puppet3` package and its dependencies:

```bash
# pkgutil -i -y puppet3
```

By default, an SMF service is created to run the Puppet agent daemonised. This is not something that we want – the updates will be run out of cron for more control and granularity (more on this later). For now, check the status of the service:

```bash
# svcxs -xv cswpuppetd
svc:/network/cswpuppetd:default (?)
State: online since November 26, 2013 08:44:35 AM EST
    See: /var/svc/log/network-cswpuppetd:default.log
Impact: None.
```

Disable it, thus stopping it also:

```bash
# svcadm disable svc:/network/cswpuppetd
```

Copy the supplied sample `puppet.conf` into place:

```bash
# cp /etc/puppet/puppet.conf.example-CSW /etc/puppet/puppet.conf
```

Update the `puppet.conf` `server` variable in the `[agent]` section as appropriate:

```bash
# vi /etc/puppet/puppet.conf
[agent]
```

Try a test run; the certificate request will be sent to the Puppet master, and can be signed as shown in the CentOS instructions above.

```bash
# puppet agent --test
```

```
Info: Creating a new SSL key for sol11test.local
Info: Caching certificate for ca
Info: Creating a new SSL certificate request for sol11test.
    local
Exiting; failed to retrieve certificate and waitforcert is disabled
```

Once the certificate is signed, a clean run should be observed:

```bash
# puppet agent --test
```

```
Info: Caching certificate_revocation_list for ca
Info: Retrieving plugin
Notice: /File[/var/opt/csw/puppet/lib]/mode: mode changed
    ‘0750’ to ‘0755’
Info: Caching catalog for sol11test.local
Info: Applying configuration version ‘1386191583’
Info: Creating state file /var/opt/csw/puppet/state/state.
    yaml
Notice: Finished catalog run in 0.05 seconds
```

A quick check of some of the `facter` variables on each type of host confirms that things are ready to go:

```bash
root@sol11test:~# facter operatingsystem
Solaris
root@sol11test:~# facter operatingsystemrelease
5.11
[root@centosa ~]# facter operatingsystem
CentOS
[root@centosa ~]# facter operatingsystemrelease
6.4
```

You can run `facter -p` to get a listing of all facts known to Puppet.

**Puppet Configuration**

The bulk of this article will now highlight some of the features of the Puppet configuration language, and how Puppet can be used to deploy security configuration to hosts.
Within an article there is an obvious limit to what can be covered, so the official Puppet documentation at http://docs.puppetlabs.com/references/latest should be consulted for authoritative information, as well as the appropriate security benchmarks for your operating system.

Before starting, it’s worth looking at the directory structure of the Puppet master installation. There are two subdirectories under /etc/puppet of note – manifests which contains Puppet manifests, and modules which contains Puppet modules. Each module is within its own subdirectory, e.g. /etc/puppet/modules/foomodule. Beneath the module subdirectory are three more directories – files (contains files you wish to serve to clients), manifests (manifests that comprise the module) and templates (any ERB templates your module uses). The entry point manifest is, by default, /etc/puppet/manifests/site.pp. Other subdirectories may exist under /etc/puppet if other features are being used (for example, hiera). Start by configuring site.pp. In this file, include any site-wide defaults. /etc/puppet/manifests/site.pp is shown in Listing 3.

A few things to notice about this example. Centralised backups to the Puppet master are configured using the filebucket type with name main and a server of centosa.local – our Puppet master. A default File object is then created causing all file modifications across all manifests and modules to be backed up to the filebucket main. .svn and .git files are ignored. We then go on to include nodes.pp via an import statement. The node declarations could go into site.pp, but we want to break things down for manageability and maintainability. nodes.pp is shown in Listing 3.

### Listing 3. /etc/puppet/manifests/site.pp

```bash
# site.pp – puppetmaster base configuration
# configure centralised backups on the puppetmaster

filebucket { main:
    path => false,
    server => 'centosa.local'
}

File {
    backup => 'main',
    ignore => ['.svn', '.git']
}

import “nodes.pp”
```

Listing 4. /etc/puppet/manifests/nodes.pp

```bash
# nodes.pp – base node configuration
# This file accomplishes two things. First, it
# forces the definition of nodes, and secondly
# it imports separate node files for each environment

node default {
    fail( “You must add a node definition for this host,
        not use the default” )
}

# node functionality broken out into separate files
import “nodes-test.pp”
```

### Listing 5. /etc/puppet/manifests/nodes-test.pp

```bash
# nodes-test.pp
# Node definition file for test nodes

# Puppetmaster
node “centosa.local” {
    include security
    include security::logging::server
}

# CentOS nodes
node “centosb.local”, “centosc.local” {
    include security
    include security::logging::client
}

# Solaris nodes
node “sol11test.local” {
    include security
    include security::logging::client
}
```

As you can see, three node definitions are present. Within these statements are included which will call the classes of configuration that will be applied. The first matches hostname centosa.local and includes two classes – the security base class (/etc/puppet/modules/security/manifests/init.pp – which we will meet again later), and the security::logging::server class. The second definition matches two CentOS nodes – centosb.local and centosc.local. We could also have used a regular expression to match these. The third declaration is for the Solaris 11 node. Both of these definitions include the base security class and the security::logging::client class.

The various classes that comprise the configuration of the nodes is defined in a Puppet module called security. Puppet looks for modules in /etc/puppet/modules by default.
Listing 6. /etc/puppet/modules/security/manifests/init.pp

```puppet
# class: security
# This is the base security class and acts as a
# wrapper around the various sub-classes.

class security {
  include security::base::files
  include security::services
  include security::tcpwrappers
  include security::kerneltuning
  include security::networktuning
  include security::sshd
  # Operating system specific
  case $::operatingsystem {
    'CentOS' : {
      security::base::filesystem { '/home' : fs => '/
        home' }
      security::base::filesystem { '/tmp' : fs => '/tmp'
        }
      security::base::filesystem { '/var/tmp' : fs => '/
        var/tmp' }
      security::base::filesystem { '/dev/shm' : fs => '/
        dev/shm' }
      include security::selinux
    }
    'Solaris' : {
      include security::coreadm
      include security::strong_iss
      include security::routeadm
    }
    default : { fail( 'OS is unsupported by security
               class' ) }
  }
}
```

Listing 7. /etc/puppet/modules/security/manifests/base/files.pp

```puppet
# class security::base::files
# Copy several base files to each node – current
# - /etc/issue
# - /etc/motd (created from template)
# - /etc/cron.* or /etc/cron.d/* depending on OS

class security::base::files { 
  case $::operatingsystem { 
    'CentOS' : {
      file { '/etc/motd' :
        ensure => 'present',
        owner => 'root',
        group => 'root',
        mode => '0644',
        content => template( 'security/motd.erb' )
      }
      file { '/etc/issue' :
        ensure => 'present',
        owner => 'root',
        group => 'root',
        mode => '0644',
        source => 'puppet:///modules/security/etc/issue'
      }
      file { '/etc/cron.allow' :
        ensure => 'present',
        owner => 'root',
        group => 'root',
        mode => '0644',
        source => 'puppet:///modules/security/etc/cron.
        allow'
      }
      file { '/etc/cron.deny' :
        ensure => 'present',
        owner => 'root',
        group => 'root',
        mode => '0644',
        source => 'puppet:///modules/security/etc/cron.
        deny'
      }
      file { '/etc/at.allow' :
        ensure => 'present',
        owner => 'root',
        group => 'root',
        mode => '0644',
        source => 'puppet:///modules/security/etc/at.
        allow'
      }
      file { '/etc/at.deny' :
        ensure => 'present',
        owner => 'root',
        group => 'root',
        mode => '0644',
        source => 'puppet:///modules/security/etc/at.
        deny'
      }
      file { '/etc/motd' :
        ensure => 'present',
        owner => 'root',
        group => 'sys',
        mode => '0644',
```
Creating a module first requires the appropriate directory structure to be in place.

```
$ mkdir -p /etc/puppet/modules/security/
    (files,manifests,templates)
```

When the bare module name is included, as is the case in our example above (include security), there should be an init.pp file at /etc/puppet/modules/<modulename>/manifests/init.pp containing the corresponding class (in our case, this would be a definition of the security class).

Let's take a look at init.pp for the security module in Listing 6. There is a lot going on here. init.pp is merely a wrapper class in this case, farming off the work to various worker classes. The first six calls (include security::base::files through to include security::sshd) are applied to all hosts. Then, a case statement evaluates the operatingsystem facter variable. Facter is installed as a Puppet prerequisite and enables Puppet to query the host for “facts” about its configuration. One such variable is operatingsystem, and this and other top-level variables can be accessed via the $::<variable_name> syntax.

Depending on whether $::operatingsystem evaluates to CentOS or Solaris will dictate what further action is taken. You can see that in the CentOS case, there are four calls to the defined type security::base::filesystem, and the inclusion of the security::selinux class. For Solaris, three additional classes are included. The default case would be evaluated should the $::operatingsystem variable contain some other value. The first included class is security::base::files.

Let's take a look at the class in Listing 7. It can be found at /etc/puppet/modules/security/manifests/base/files.pp. Note that the subdirectory base has been created and this matches the class name (using :: as a path separator).

Again, the content of the $::operatingsystem variable is evaluated and the appropriate file types are defined. Let's break down a couple of the entries. First, /etc/issue.

Here, the file /etc/issue will be created if it doesn't exist (we ensure that it's present), and the owner will be set to root, the group to root, and the permissions to 0644. The source of the file is on the Puppet master.
Securing CentOS and Solaris 11 with Puppet

at puppet:///modules/security/etc/issue (which corresponds to a physical file path of /etc/puppet/modules/security/files/etc/issue). There are many more configuration attributes for the file type; consult the type reference (http://docs.puppetlabs.com/references/latest/type.html) for further detail.

Another type of entry is shown below, making use of ERB templates, here for Solaris hosts:

```puppet
file { '/etc/motd' :
  ensure => 'present',
  owner => 'root',
}
```

```
Listing 8. /etc/puppet/modules/security/manifests/services.pp

# class: security::services
# Takes care of ensuring unneeded services are
# not running and are disabled
class security::services {

case $::operatingsystem {
  'CentOS' : {
    # services netconsole, rdisc and saslauthd are
    # shipped
    # disabled with CentOS 6.x minimal, let’s keep
    them that
    # way
    service { 'netconsole' :
      enable => false,
      ensure => stopped
    }
    service { 'rdisc' :
      enable => false,
      ensure => stopped
    }
    service { 'saslauthd' :
      enable => false,
      ensure => stopped
    }
  }
  'Solaris' : {
    # GDM not fully installed on a text-mode
    # installation of
    # Solaris 11, so we can comment it here to save
    puppet
    # complaining about an unmanageable service state
    # service { 'svc:/application/graphical-login/gdm' :
    #   enable => false,
    #   ensure => stopped
    # }
    service { 'svc:/network/keyserv' :
      enable => false,
      ensure => stopped
    } default : {
      fail( 'OS unsupported by security::services class'
    }
    service { 'svc:/network/nis/server' :
  }
```
As you can see, the `security/motd.erb` template is being used to populate the file (the `content` attribute). The physical path to the file is `/etc/puppet/modules/security/templates/motd.erb`. Here is the ERB file:

```erb
Welcome to <%= fqdn %>. Authorised users only. All activity may be monitored and reported.
```

`fqdn` is a [facter](http://www.facter.org/) variable, but we could equally reference any variable that’s defined in the appropriate scope, for example in the calling class.

A more complete example using ERB templates to configure Apache VirtualHosts can be found on my website ([http://www.tokiwinter.com/puppet-module-apache2-virtual-host-templates](http://www.tokiwinter.com/puppet-module-apache2-virtual-host-templates)). We can now look at the next class – `security::services` – in Listing 8.

This class takes care of stopping any unnecessary services, and disabling them via whatever OS-specific mechanism is required (which Puppet hides from us, whether the provider is `init.d` scripts or SMF, we have a consistent interface via the service type).

The next class, `security::tcpwrappers`, is quite complex. It uses Hiera to look up variable values in a hierarchical database, in our case stored as plain text in YAML format. Hiera used to be an additional tool, but has been integrated fully with Puppet since version 3.0. Let’s start with the class definition – take note of the positional parameters `$hostsallow` and `$hostsdeny`. These two variables will be populated via a Hiera lookup. Since version 3.x, automatic parameter lookup in Hiera is enabled by default (see [http://docs.puppetlabs.com/hiera/1/puppet.html#automatic-parameter-lookup](http://docs.puppetlabs.com/hiera/1/puppet.html#automatic-parameter-lookup) for more details).

### Listing 9. `/etc/puppet/modules/security/manifests/tcpwrappers.pp`

```puppet
# class security::tcpwrappers
# Set up tcpwrappers on Solaris and CentOS hosts
class security::tcpwrappers ( $hostsallow = "", $hostsdeny = "" ) {
  file { '/etc/hosts.allow' :
    owner  => 'root',
    group  => 'root',
    mode   => '0644',
    source => $hostsallow
  }
  file { '/etc/hosts.deny' :
    owner  => 'root',
    group  => 'root',
    mode   => '0644',
    source => $hostsdeny
  }
  case $::operatingsystem {
    'CentOS' : {
      # nothing else to do, files are in place
    }
    'Solaris' : {
      # set up inetd-controlled services for tcp_wrappers
      exec { '/usr/sbin/inetadm -M tcp_wrappers=TRUE' :
        unless => '/usr/sbin/inetadm -p | /bin/grep tcp_wrappers=TRUE'
      }
    }
    default : { fail( 'OS unsupported by security::tcp_wrappers class' ) }
  }
}
```

### Listing 10. `/etc/puppet/hiera.yaml`

```yaml
---
:backends:
  - yaml
:yaml:
  :datadir: /etc/puppet/hieradata
:hierarchy:
  - %(::clientcert)
  - %(::operatingsystem)
  - common
```
Therefore, the values of `security::tcpwrappers::hostsallow` and `security::tcpwrappers::hostsdeny` will be looked up in Hiera.

The Hiera configuration is defined on the Puppet master at `/etc/puppet/hiera.yaml` as a plain text YAML file. Its contents are shown in Listing 10.

So that the `hiera` command-line utility works as expected, remove the installed `/etc/hiera.yaml` and symlink:

```
# rm -f /etc/hiera.yaml
# ln -s /etc/puppet/hiera.yaml /etc
```

The Hiera configuration above does several important things. Firstly, it defines the available `:backends:` – here we use the `yaml` backend. The `:datadir:` of `/etc/puppet/hieradata` for the `:yaml:` files is defined next, followed by the `:hierarchy:` we wish to use. Our `:hierarchy:` is as follows:

```
:hierarchy:
  - %{::clientcert}
  - %{::operatingsystem}
  - common
```

First, the `clientcert` facter variable is checked, which will return the common name of the client certificate – generally the fully-qualified domain name of the host. If the file `/etc/puppet/hieradata/%{::clientcert}.yaml` exists, the `security::tcpwrappers::hostsallow` and `security::tcpwrappers::hostsdeny` variables looked up within them. If that file doesn’t exist, the `operatingsystem` variable is checked. If `/etc/puppet/hieradata/%{::operatingsystem}.yaml` exists, the variables are looked up there, and finally the catch all – if the granular tests fail, `common.yaml` will be used.

The contents of the YAML files are as follows:

```
# cat centosa.local.yaml
---
security::tcpwrappers::hostsallow: "puppet:///modules/security/etc/hosts.allow-centosa.local"
security::tcpwrappers::hostsdeny: "puppet:///modules/security/etc/hosts.deny-centosa.local"
# cat common.yaml
---
security::tcpwrappers::hostsallow: "puppet:///modules/security/etc/hosts.allow-common"
security::tcpwrappers::hostsdeny: "puppet:///modules/security/etc/hosts.deny-common"
# cat Solaris.yaml
---
```

www.bsdmag.org
Host `centosa.local` would use `centosa.local.yaml` (due to `%(::clientcert)` in the hierarchy) and pull the values in for `security::tcpwrappers::hostsallow` and `security::tcpwrappers::hostsdeny` from that file. Host `centosb.local` would fall through to `common.yaml` (unless there was a `%(::clientcert).yaml` or `CentOS.yaml`), and a Solaris host would use `Solaris.yaml`. Looking back at the `security::tcpwrappers` class, you can see that this substitution occurs during class instantiation as positional parameters:

```erlang
class security::tcpwrappers { $hostsallow = "", $hostsdeny = "" } {
...
}
```

These variables are then referenced in the file type definitions:

```erlang
source -> $hostsallow
```

These variables will contain the output of the `hiera` look-up and thus the source attribute will reference the correct substitution.

---

**Listing 11. /etc/puppet/modules/security/manifests/ipadm.pp**

```erlang
# defined type: security::ipadm
# Use ipadm to check the value of an ipadm-controlled
# variable
# and update it if required.
#
# Parameters:
#  - $variable => the ipadm variable we want to check/
#                 change
#  - $co       => should be set to "current"
#  - $protocol => the protocol (ip, tcp, etc.)
#  - $value    => the value to check and/or set
define security::ipadm( $variable = '', $co = '',
  $protocol = '', $value = '' ) {
  exec { "/usr/sbin/ipadm set-prop -p
  ${variable}=${value} ${protocol}"
    # ensure => '/usr/sbin/ipadm show-prop -p
    #         ${variable} -co ${co} ${protocol} | /bin/grep '^${value}$'
  }
}
```

**Listing 12. /etc/puppet/modules/security/manifests/kerneltuning.pp**

```erlang
# class: security::kerneltuning
# Updates the following files:
# CentOS
#  - /etc/security/limits.conf
#  - /etc/sysctl.conf (and calls sysctl -p if required)
# Solaris
#  - /etc/system

class security::kerneltuning {
  case $::operatingsystem {
    'CentOS' : {
      file { "/etc/security/limits.conf" :
        ensure => 'present',
        owner  => 'root',
        group  => 'root',
        mode   => '0644',
        source => 'puppet:///modules/security/etc/security/limits.conf'
      }
    }
    'Solaris' : {
      file { "/etc/system" :
        ensure => 'present',
        owner  => 'root',
        group  => 'sys',
        mode   => '0644',
        source => 'puppet:///modules/security/etc/system'
      }
    }
  }
  default : { fail( 'OS unsupported by
               security::kerneltuning class' ) }
}
```
version of the file for the node we are deploying to. After the files are deployed as appropriate, Solaris has some additional checks and/or configuration performed. First, `inetadm -p` is checked for `tcp_wrappers=TRUE` to verify whether `inetd`-controlled services are configured to use TCP Wrappers. If not set to TRUE, `inetadm -M` is used to update the configuration. Next, TCP Wrappers is enabled for the RPC portmapping service if it isn’t already via a call to `svccfg`. The service type definition is required to give us something to notify from the `exec`. Defined types are pieces of code you want to call repeatedly with different parameters, akin to functions. `security::ipadm` is a defined type to check and set properties on Solaris hosts with `ipadm`. Its contents are shown in Listing 11.

This can then be called from classes. The commented documentation in the listing explains each of the parameters. They default to empty, which would cause the commands to syntax error. You should add checks that sanitise input via conditionals. We will call this a defined type in later manifests.

Next, we deploy kernel tuning changes – again the file is well commented. On Solaris, a change to `/etc/system

---

**Listing 13. `/etc/puppet/modules/security/manifests/networktuning.pp`**

```perl
# class: security::networktuning
# Configure Solaris IP stack via ipadm
# Requires the security::ipadm defined type.

class security::networktuning {
    case $::operatingsystem {
        'CentOS': {
            # On CentOS, this is all taken care of in
ekerneltuning.pp
        }
        'Solaris': {
            security::ipadm{ '_respond_to_timestamp':
                variable => '_respond_to_timestamp',
                current, protocol => 'ip', value => '0'
            }
            security::ipadm{ '_forward_directed_broadcasts':
                variable => '_forward_directed_broadcasts',
                current, protocol => 'ip', value => '0'
            }
            security::ipadm{ '_respond_to_timestamp_broadcast':
                variable => '_respond_to_timestamp_broadcast',
                current, protocol => 'ip', value => '0'
            }
            security::ipadm{ '_respond_to_address_mask':
                variable => '_respond_to_address_mask',
                current, protocol => 'ip', value => '0'
            }
            security::ipadm{ '_respond_to_echo_broadcast':
                variable => '_respond_to_echo_broadcast',
                current, protocol => 'ip', value => '0'
            }
            default: { fail( 'OS unsupported by
                security::kerneltuning class' ) }
        }
    }
}

**Listing 14. `/etc/puppet/modules/security/manifests/sshd.pp`**

```perl
class security::sshd {
    case $::operatingsystem {
        'CentOS': {
            package { [ 'openssh', 'openssh-clients',
                'openssh-server' ]:
                ensure => 'latest'
            }
            file { '/etc/ssh/sshd_config':
                owner => 'root',
                group => 'root',
                mode => '0600',
                source => 'puppet:///modules/security/etc/ssh/
sshd_config-centos',
                require => Package[ "openssh-server" ],
                notify => Service[ "sshd" ]
            }
            service { 'sshd':
                require => File[ "/etc/ssh/sshd_config" ],
                ensure => 'running',
                enable => 'true'
            }
        }
        'Solaris': {
            file { '/etc/ssh/sshd_config':
                owner => 'root',
                group => 'sys',
                mode => '0644',
                source => 'puppet:///modules/security/etc/ssh/
sshd_config-solaris',
                notify => Service[ "svc:/network/ssh" ]
            }
            service { 'svc:/network/ssh':
                enable => 'true',
                ensure => 'running'
            }
        }
    }
}
```
requires a reboot – obviously we don’t orchestrate that ...
For suggested values for these files, see my previous
articles published in PenTest Magazine on Securing the
Linux and Solaris 11 Operating Systems.
All is very straightforward in the above class. Files are
copied for both hosts, and on CentOS, the sysctl -p
command is run only if the /etc/sysctl.conf file changes
(refreshonly-true makes the exec respond to events,
and for that we use a file subscription to /etc/sysctl.
conf). security::networktuning is the next class, and
it shows how the security::ipadm defined type can be
called: Listing 12 and Listing 13.

Listing 15. /etc/puppet/modules/security/manifests/base/
filesystem.pp

```perl
Listing 16. /etc/puppet/modules/security/manifests/selinux.pp
```

```bash
define security::base::filesystem ( $fs = '' ) {
case $::operatingsystem {
  'CentOS' : { }
default : { fail( 'OS not supported by the
    security::base::filesystem class' ) }
}
if $fs == '' {
  fail('No FS passed')
} else {
  case $fs {
    '/tmp', '/var/tmp', '/dev/shm' : {
      case $fs {
        '/tmp' : { $lv = '\/dev\/mapper\/vg_sys-lv_
tmp' }
        '/var/tmp' : { $lv = '\/dev\/mapper\/vg_
sys-lv_var_tmp' }
        '/dev/shm' : { $lv = 'tmpfs' }
        default : { fail('Fs not implemented in
          security::base::filesystem') }
      }$mountopts = "noexec,nosuid,nodev"
exec { "/bin/sed -i '/^$lv/ s/defaults/
defaults,$mountopts/' /etc/fstab" :
  unless => "/bin/grep '^$lv[[[:space:]]].*default
s,$mountopts' /etc/fstab"
}
$mountcommand = "/bin/mount -o remount,$mountopts
$fs"
exec { "$mountcommand" :
  unless => "/bin/grep
''$lv[[[:space:]]].*$mountopts' /etc/mtab"
}
default : { fail('Fs not supported by security::base::filesystem') }
}
}

Listing 16. /etc/puppet/modules/security/manifests/selinux.pp
```

```bash
class security::selinux {
case $::operatingsystem {
  'CentOS' : {
    file { '/etc/selinux/config' :
      owner => 'root',
      group => 'root',
      mode  => '0644',
      source => 'puppet:///modules/security/etc/
      selinux/config'
    }
    exec { '/usr/sbin/setenforce Enforcing' :
      unless => '/usr/sbin/getenforce | /bin/grep
      Enforcing'
    }
  }
default : { fail('OS unsupported by
    security::selinux class') }
}
```
Securing CentOS and Solaris 11 with Puppet

Listing 17. /etc/puppet/modules/security/manifests/coreadm.pp

```puppet
# class: security::coreadm
# Ensure core dumps are disabled on Solaris hosts
class security::coreadm {
case $::operatingsystem {
  'Solaris' : {
    exec { '/usr/bin/coreadm -d global -d process -d global-setid -d proc-setid -d log' :
      onlyif => '/usr/bin/coreadm | /bin/grep enabled'
    }
  }
  default : { fail( 'OS unsupported by security::coreadm class' ) }
}
}
```

Listing 18. /etc/puppet/modules/security/manifests/coreadm.pp

```puppet
# class: security::routeadm
# Ensure routing is disabled on our Solaris hosts – we don’t use them for routing
class security::routeadm {
case $::operatingsystem {
  'Solaris' : {
    exec { '/usr/sbin/routeadm -d ipv4-routing -d ipv6-routing -d ipv4-forwarding -d ipv6-forwarding; /
      /usr/sbin/routeadm -u' :
      onlyif => '/usr/sbin/routeadm -p | /bin/grep enabled'
    }
  }
  default : { fail( 'OS unsupported by security::routeadm class' ) }
}
}
```

Listing 19. /etc/puppet/modules/security/manifests/strong_iss.pp

```puppet
# class: security::strong_iss
# Enforce strong TCP initial sequence number generation
class security::strong_iss {
case $::operatingsystem {
  'Solaris' : {
    file { '/etc/default/inetinit' :
      source => 'puppet:///modules/security/etc/default/inetinit'
    }
    security::ipadm{ "strong_iss" : variable => '_strong_iss', co => 'current', protocol => 'tcp',
      value => '2' }
    default : { fail( 'OS unsupported by security::coreadm class' ) }
  }
  default : { fail( 'OS unsupported by security::routeadm class' ) }
}
}
```

Listing 20. Running the Puppet Agent

```plaintext
# puppet agent --test
Info: Retrieving plugin
Info: Caching catalog for centosa.local
Info: Applying configuration version '1387232042'
Notice: /File[/etc/ssh/sshd_config]/content: --- /etc/ssh/sshd_config 2013-12-05 21:29:51.886986435 +1100
+++ /tmp/puppet-file20131217-48343-1hbek4h-0 2013-12-17 09:15:08.874975252 +1100
@@ -128,7 +128,7 @@
    #ChrootDirectory none
    # no default banner path
    +Banner /etc/issue
    # override default of no subsystems
    Subsystem sftp /usr/libexec/openssh/sftp-server
    Info: /File[/etc/ssh/sshd_config]: Filebucketed /etc/ssh/sshd_config to main with sum 226c398f540ca2322ffa01e4cf2c3646
    Notice: /File[/etc/ssh/sshd_config]: content changed '{md5}226c398f540ca2322ffa01e4cf2c3646' to '{md5}29a87b64a0f815035a8bde658bc01504'
    Info: /File[/etc/ssh/sshd_config]: Scheduling refresh of Service[sshd]
    Notice: /Stage[main]/Security::Sshd/Service[sshd]: Triggered 'refresh' from 1 events
    Notice: /File[/var/syslog]/seltype: seltype changed 'var_log_t' to 'var_t'
    Notice: /Stage[main]/Security::Logging::Server/Exec[/usr/bin/chcon --reference=/var/log /var/syslog]/returns: executed successfully
    Notice: Finished catalog run in 41.36 seconds
```
security::sshd is shown in Listing 14. On CentOS hosts, it checks that the appropriate packages are installed and up-to-date. For both OSes, it then copies an appropriate sshd_config into place, before enabling and starting the service. If the configuration file is changed, the service is notified and refreshed. Going back to init.pp, we have now discussed all of the OS-generic classes. Next, the OS-specific classes are considered:

```perl
case $::operatingsystem {
  'CentOS' : {
    security::base::filesystem { '/home' : fs => '/home' }
    security::base::filesystem { '/tmp' : fs => '/tmp' }
    security::base::filesystem { '/var/tmp' : fs => '/var/tmp' }
    security::base::filesystem { '/dev/shm' : fs => '/dev/shm' }
    include security::selinux
  }
  'Solaris' : {
    include security::coreadm
    include security::strong_iss
    include security::routeadm
  }
}
```

You’ll note another defined type – security::base::filesystem. It is shown in Listing 15. This code will ensure that appropriate secure mount options are used in both /etc/fstab, and for any current live mounts via a call to mount --remount where necessary. The final CentOS specific class is security::selinux, which checks that SELinux is configured and enforcing. It is shown in Listing 16. There are three final classes to discuss – the Solaris-specific classes. The first two, security::coreadm (Listing 17) and security::routeadm (Listing 18) are very straightforward. They check whether core dumps and routing are enabled, respectively, and disable the functionality if it is. You may need to adjust this to suit your purposes and site policy. The final class, security::strong_tss is shown in Listing 19. You’ll note it calls the security::ipadm defined type. This has only scratched the surface of what Puppet can do. There are further examples of Puppet modules and configuration on my website, as well as articles on adding robustness to your Puppet master (replacing the built-in webserver with Apache and Passenger), and integrating Puppet with a git workflow. This article does not cover all security aspects of hardening CentOS and Solaris hosts. It serves as a guide to show you the power of Puppet, and set you writing your own modules and custom defined types.

You’ll see a lot of changes take place. For the sake of brevity, here is a brief excerpt from a run: Listing 20.

Run again and you should have a clean run as the configuration is up-to-date. You can daemonise the Puppet agent, however I prefer to run out of cron. You can provide granular timing via cron, and control which hosts fetch their configuration at which times to balance load. I prefer to run puppet agent --test and have that output to a log file which is then managed by logrotate so that we have logs of verbose agent output. You may need to adjust this to suit your needs. Add a cron job for root such as the following:

```
06,36 * * * * /path/to/puppet agent --test >>/var/log/puppet.log 2>&1
```

This would run the agent at 06 and 36 past the hour, outputting both STDOUT and STDERR to /var/log/puppet.log. You can then manage this log via logrotate. Read the manual page for logrotate.conf for details.

As a final noteworthy point, you can view a log file of all HTTP requests made to the server by checking the masterhttp log, which is located at /var/log/puppet/masterhttp.log by default.

**Conclusion**

This has only scratched the surface of what Puppet can do. There are further examples of Puppet modules and configuration on my website, as well as articles on adding robustness to your Puppet master (replacing the built-in webserver with Apache and Passenger), and integrating Puppet with a git workflow. This article does not cover all security aspects of hardening CentOS and Solaris hosts. It serves as a guide to show you the power of Puppet, and set you writing your own modules and custom defined types.

---

**TOKI WINTER**

Toki Winter is an experienced UNIX/Linux System Administrator with over a decade of experience managing large enterprises and running many thousands of services. His website, http://www.tokiwinter.com, contains many useful articles, HOWTOs and tips for the advanced UNIX administrator.
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 DELIVER 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
User, Group and Password Management on Linux and Solaris

This article will cover the user, group and password management tools available on the Linux and Solaris Operating Systems. The specific versions covered here are CentOS 6.4 and Solaris 11.1, though the commands will transfer to many other distributions without modifications (especially RHEL and its clones), or with slight alterations to command options. Check your system documentation and manual pages for further information.

What you will learn…
• How to manage users effectively and securely

What you should know…
• Basic Unix knowledge.

Knowing how to manage users effectively and securely is a requirement of financial standards such as PCI-DSS, and information security management systems such as ISO 27001.

In this article, I will consider local users and groups – coverage of naming services such as NIS and LDAP is beyond its scope but may be covered in a future article.

This article also presumes some prior basic system administration exposure with a UNIX-like operating system.

Users and Groups

Users and groups make up a fundamental part of any multi-user operating system. On UNIX and Linux systems, every user has a UID (User ID) and a primary GID (Group ID). Users can own files, use resources and execute processes. The System Administrator can grant access to resources and data based upon the UID of the user, as well as the groups a user is a member of. Some additional features such as RBAC in Solaris take things a step further and allow very fine-grained control of what a user can and cannot do/access. The OS takes care of mapping usernames and group names to UIDs and GIDs and vice versa by using a naming service, such as file-based (described next), LDAP or NIS.

Using the standard file-based naming service, the main user database is /etc/passwd. The group database is at /etc/group, and the shadow password file at /etc/shadow. Linux systems also have a group shadow database at /etc/gshadow.

Why do we need a shadow password file? /etc/passwd needs to be readable by every user on the system, as some applications depend on being able to map UIDs to...
User, Group and Password Management on Linux and Solaris

Usernames (ls, for example) and thus need to be able to read the password database. Therefore, the encoded password is moved to the /etc/shadow file, which need only be readable by root. Commands such as /usr/bin/passwd are SUID (Set UID) root so they can be executed by normal users with root privileges, thus being able to update the shadow file.

```
# ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 30768 Feb 22 2012 /usr/bin/passwd
```

UIDs and GIDs can be duplicated, but it is bad practice, as it makes auditing harder and could lead to data being revealed inadvertently.

Let’s start by taking a look at the format of the various password database files.

/etc/passwd

The format of this file is:

```
username:x:uid:gid:GECOS:home:shell
```

The first field contains the user’s username. The second field contains an “x” which indicates that the shadow password suite is in use. The third is the UID, which, as discussed, should be unique. The fourth is the users’ primary group’s GID. The fifth field is known as the GECOS field, after the GE operating system of the same name, and typically contains the user’s full name, and perhaps other identifying information; this is the only field in the file that may contain a space. The sixth field is the user’s home directory – an absolute path. The final field is the absolute path to the user’s login shell – which should be specified in /etc/shells on a Linux system. Solaris doesn’t use /etc/shells.

```
# cat /etc/redhat-release
CentOS release 6.4 (Final)
# cat /etc/shells
/bin/sh
/bin/bash
/sbin/nologin
/bin/bash
```

/etc/shadow

The format of this file is:

```
```

The first field is the username and should correspond to an entry in /etc/passwd. The encoded password comes next. last changed is the number of days after Jan 1, 1970 that the password was last changed – set to 0 it will force a user to change their password upon next login. mindays is the minimum password age, which is the number of days the user will have to wait before they will be allowed to change their password again. maxdays is the maximum password age, which is the number of days after which the user will have to change their password again. warn is the password warning period, which is the number of days before a password is going to expire during which the user should be warned. inactive is the password inactivity period – the number of days after a password has expired during which the password will still be accepted. expire is the account expiration date – the date of expiration of the account, expressed as the number of days since Jan 1, 1970. The final field is reserved for future use. On an out-of-the-box CentOS install, many fields are empty when users are added via useradd without additional password ageing options:

```
# fgrep toki /etc/shadow
toki:encoded_password_here:16041:0:99999:7:::
```

Again, this is default behaviour that we will learn to configure over the course of this article.
/etc/group

The format of the file is:

group_name:passwd:GID:user_list

group_name is the name of the group. passwd is the encoded group password. The Solaris manual page for group(4) states “Group passwords are antiquated and not often used.” The GID is the numerical group ID, and the user_list is a comma-separated list of group members. There could also be users having this group as their primary GID (i.e. the GID specified in /etc/passwd), in which case they wouldn’t need to appear in user_list. Here are two entries from /etc/group on a CentOS machine:

# fgrep “toki” /etc/group
wheel:x:10:toki
toki:x:333:

and here are two from a Solaris 11 machine:

# fgrep “sys” /etc/group
sys::3:root,bin,adm
sysadmin::14:

On a Solaris machine, the passwd field is empty for all groups out-of-the-box. On a Linux machine, it contains an “x” indicating that the group shadow file is in use. See man 5 gshadow for more information on the group shadow file (/etc/gshadow), but its format is:

username:encoded_password:administrators:members

As you can see, I’m a member of the wheel group on this machine:

# fgrep toki /etc/gshadow
wheel:::toki
toki:!!!:

The user administration tools and group administration tools will take care of updating these files for us, and keeping /etc/shadow in sync with /etc/passwd, and /etc/gshadow with /etc/group.

Adding Users

Let’s start by taking a look at how the two OSes create users using all default and no additional arguments other than the required username. On CentOS:

# useradd testuser1
# fgrep testuser1: /etc/passwd
testuser1:x:334:334::/home/testuser1:/bin/bash
# fgrep testuser1: /etc/group
testuser1:x:334:

It took the next available UID on my system, and applied sensible defaults to the other parameters (a home of /home/<username>, shell of /bin/bash). It also created a group, and set it to be the primary group of the new user. This new-group-per-user policy can be found on all RHEL-derivatives. On Solaris, the behaviour is slightly different:

# useradd tstusr01
# fgrep tstusr01 /etc/passwd
tstusr01:x:102:10::/export/home/tstusr01:/usr/bin/bash
# fgrep tstusr01 /etc/group

As you can see, the home directories and shell path are different (and are appropriate for this OS) but the account is just added to a group called staff:

# awk -F: ‘$3 == 10 { print $0 }’ /etc/group
staff:::10:

On CentOS, you will note that the user’s home directory has been created, the contents of /etc/skel copied in, and then appropriate ownership and permissions configured. /etc/skel is a good way to push site-wide configuration files out to all new accounts. As an example, here are the contents of /etc/skel on a CentOS system. A mail spool file has also been created:

# ls -l /etc/skel
total 12
-rw-r--r--. 1 root root  18 Feb 22  2013 .bash_logout
-rw-r--r--. 1 root root 176 Feb 22  2013 .bash_profile
-rw-r--r--. 1 root root 124 Feb 22  2013 .bashrc
# ls -l /home/testuser1
total 12
-rw-r--r--. 1 testuser1 testuser1  18 Feb 22  2013 .bash_logout
-rw-r--r--. 1 testuser1 testuser1 176 Feb 22  2013 .bash_profile
-rw-r--r--. 1 testuser1 testuser1 124 Feb 22  2013 .bashrc
# ls -l /var/spool/mail/testuser1
-rw-rw----. 1 testuser1 mail 0 Dec  4 05:39 /var/spool/mail/testuser1
On Solaris, by default, the user’s home directory is not created, and the contents of /etc/skel are not, therefore, copied in. We can either do this manually after running `useradd`, or instead use options available to the `useradd` command to create the user. We can use the `-m` option to cause the home directory to be created, and the `-d` option to explicitly define the home directory location.

We then observe the files being copied in from /etc/skel:

```
# useradd -m -d /export/home/tstusr02 tstusr02
80 blocks
# ls -1A /export/home/tstusr02
  total 11
-r--r--r-- 1 tstusr02 staff 159 Nov 25 11:09 .bashrc
-rw-r--r-- 1 tstusr02 staff 568 Nov 25 11:09 .profile
-rw-r--r-- 1 tstusr02 staff 166 Nov 25 11:09 local.cshrc
-rw-r--r-- 1 tstusr02 staff 170 Nov 25 11:09 local.login
-rw-r--r-- 1 tstusr02 staff 131 Nov 25 11:09 local.profile
```

We observe the files being copied in from /etc/skel:

```
# ls -1A /etc/skel
  total 11
-r--r--r-- 1 root    bin   159 Sep 20  2012 .bashrc
-rw-r--r-- 1 root    other  568 Sep 20  2012 .profile
-rw-r--r-- 1 root    sys   166 Sep 20  2012 local.cshrc
-rw-r--r-- 1 root    sys   170 Sep 20  2012 local.login
-rw-r--r-- 1 root    sys   131 Sep 20  2012 local.profile
```

The `useradd` command is very rich in terms of options and customisation. The following command adds a user `jsmith` with UID 450, a primary group of `wheel`, a home directory of `/users/jsmith`, and a login shell of `zsh`:

```
# useradd -m -d /users/jsmith -u 450 -g wheel -c "John Smith" -s /bin/zsh jsmith
```

These options can be used to override defaults. All user additions should be logged, and whether that is by using the auditing features available with your operating system, or some other logging process, will depend on your needs.

**Defaults**

When configuring new user accounts with `useradd`, there are some defaults that are used. On a CentOS system, these are by default read from `/etc/login.defs`. Here are the variables in use on a default CentOS 6.4 installation, which will need to be tuned according to the needs of your site or organisation, and any security policies in effect. There are other variables in the file, and the manual page should be consulted for further information. The file itself is also very well commented.

```
# grep "^\[^#\]\" /etc/login.defs
MAIL_DIR /var/spool/mail
```

The defined variables are as follows:

- **MAIL_DIR** – The directory where mailboxes reside.
- **PASS_MAX_DAYS** – Maximum number of days a password may be used.
- **PASS_MIN_DAYS** – Minimum number of days allowed between password changes (0 to disable).
- **PASS_MIN_LEN** – Minimum acceptable password length.
- **PASS_WARN_AGE** – Number of days warning given before a password expires.
- **UID_MIN/MAX** – Min/max values for automatic UID selection in useradd.
- **GID_MIN/MAX** – Min/max values for automatic GID selection in groupadd.
- **USERDEL_CMD** – Commented by default. Can be used to set up a customised local userdel, to remove at/cron/print jobs, etc.
- **CREATE_HOME** – Set to yes if useradd should create home directories for users by default.
- **UMASK** – Permission mask to be used.
- **USERGROUPS_ENAB** – This enables userdel to remove user groups if no members exist.
- **ENCRYPT_METHOD** – Encryption method used to encrypt passwords – we use SHA512.

**User Modification**

The `usermod` command is used to modify user accounts. The options are similar to those supplied by the `useradd` command.

For example, to move the home directory for user `tstusr02` from `/export/home/tstusr02` to `/users/tstusr02`, we could issue the following command:

```
# usermod -m -d /users/tstusr02 tstusr02
```

To change the UID for the `sometest` user to 1234 we’d issue:

```
# usermod -u 1234 sometest
```
Add the user toki to an additional secondary group (-a for “append” – not available on Solaris):

```
# usermod -a -G sysadmin toki
```

### User Deletion

Users can be deleted using the `userdel` command. There is an `-r` option that will remove the user’s home directory. It is advised that accounts are not deleted, rather just locked (see the Locking Accounts section of this article). This provides auditing capabilities, and stops UID reuse (and inadvertently giving access to an old user’s files to a new user). To delete user toki from the system, including removal of the user’s home directory:

```
# userdel -r toki
```

### Group Administration

Groups are administered with the `groupadd`, `groupmod` and `groupdel` commands. To add a new group `tstgrp` to the system, with GID 250, issue:

```
# groupadd -g 250 tstgrp
```

To change the name of `tstgrp` to `testgrp`:

```
# groupmod -n testgrp tstgrp
```

To remove a group:

```
# groupdel testgrp
```

### Password Management

Passwords are managed with the `passwd` command. An initial password can be set for a user (or a password reset) via:

```
# passwd <username>
```

A user may reset his own password with a simple:

```
$ passwd
```

which will then prompt them for the old password, followed by the new password, followed by the new password again to verify. `root` can change passwords without being prompted for the old password.

The `passwd` command can also be used to lock accounts (see the Locking Accounts section below).

### Password Defaults

On Solaris, `/etc/default/passwd` can be used to configure password-complexity enforcement. Out of the box, the `/etc/default/passwd` file contains the following defined variables:

```
# grep `"[^#]"` /etc/default/passwd
```

And the following commented variables:

```
# grep `^[A-Z]` /etc/default/passwd
```

These values can be modified according to your site’s security policy and affect the way that the `passwd` command works, and how it enforces password complexity and reuse. Again, the `/etc/default/passwd` file is well commented and the variable names themselves are self-explanatory.

### Locking Accounts

The `passwd` command is used to lock accounts on both Linux and Solaris. To lock the password for `tstusr01` on Solaris:

```
# passwd -l tstusr01
Password information changed for tstusr01
```

Looking at `/etc/shadow`, you’ll see that the string `*LK*` has been prepended to the encrypted password field in the second field:

```
# fgrep tstusr01 /etc/shadow
tstusr01:*LK*<encrypted password>:16034::::::4320
```

Doing the same on CentOS 6.4 for user `testuser1`: 

```
# grep ^tstusr01 /etc/shadow
```
User, Group and Password Management on Linux and Solaris

```bash
# passwd -l testuser1
Locking password for user testuser1.
passwd: Success
# fgrep testuser1 /etc/shadow
testuser1:!!<encrypted password>:16043:0:99999:7:::
```

You can see that `!!` has been prepended. To unlock the account on Solaris, use `passwd -u`:

```bash
# passwd -u tstusr01
passwd: password information changed for tstusr01
# fgrep tstusr01 /etc/shadow
tstusr01:<encrypted password>:16034::::::4768
```

The same applies to Linux:

```bash
# passwd -u testuser1
Unlocking password for user testuser1.
passwd: Success
# fgrep testuser1 /etc/shadow
testuser1:<encrypted password>:16043:0:99999:7:::
```

### Configuring Password Ageing

In the following example, I'll bring everything together. A group called `testusrs` with members `test01` and `test02` will be created, and various operations performed with respect to password configuration.

We'll start with Solaris. Create the group, followed by the two users:

```bash
# groupadd -g 1010 testgrp
# useradd -m -d /export/home/test01 -s /bin/bash -c "Test User 01" -u 1010 -g testgrp test01
# useradd -m -d /export/home/test02 -s /bin/bash -c "Test User 02" -u 1011 -g testgrp test02
```

Set an initial password on the two accounts:

```bash
# passwd test01
New Password:
Re-enter new Password:
passwd: password successfully changed for test01
# passwd test02
New Password:
Re-enter new Password:
passwd: password successfully changed for test02
```

Next, use `passwd -f` to force the users to change their passwords at login time:

```bash
# passwd -f test01
# passwd -f test02
```

Let’s implement some password ageing controls. Suppose the site-wide password policy is as follows:

- The minimum number of days required between password changes is 7
- The maximum number of days the password is valid for is 28
- The user will receive warnings 7 days before expiry
- Passwords will be 8 characters or more long
- A history of 5 passwords will be kept and prevented from reuse

To do this, edit `/etc/default/password` and update the following variables:

```bash
# vi /etc/default/password
# grep '^[^#]' /etc/default/passwd
MAXWEEKS=1
MINWEEKS=4
PASSLENGTH=8
HISTORY=5
```

And update any already existing user accounts:

```bash
# passwd -n 7 -x 28 -w 7 test01
passwd: password information changed for test01
# passwd -n 7 -x 28 -w 7 test02
passwd: password information changed for test02
```

Looking at `/etc/shadow` we can see the password ageing fields updated:

```bash
# grep `test0[12]` /etc/shadow
test01:<encrypted password>:16034:7:28:7:::4976
test02:<encrypted password>:16034:7:28:7:::4976
```

Let’s perform the same steps on a CentOS system. Start with the group and user creation:

```bash
# groupadd -g 1010 testgrp
# useradd -m -d /home/test01 -s /bin/bash -c "Test User 01" -u 1010 -g testgrp test01
# useradd -m -d /home/test02 -s /bin/bash -c "Test User 02" -u 1011 -g testgrp test02
```

Set the initial passwords:

```bash
# passwd test01
```
Changing password for user test01.
New password: 
Retype new password:

```
passwd: all authentication tokens updated successfully.
# passwd test02
Changing password for user test02.
New password: 
Retype new password:
```

passwd: all authentication tokens updated successfully.

On Linux, password ageing is the domain of the `chage` command. Setting the number of days since January 1st, 1970 when the password was last changed to 0 has the same effect as `passwd -f` on Solaris.

```
# chage -d 0 test01
# chage -d 0 test02
```

To configure the same password ageing policy as the Solaris example, use `chage` as follows:

```
# chage -m 7 -M 28 -W 7 test01
# chage -m 7 -M 28 -W 7 test02
# grep '^test0\[12\]:' /etc/shadow
test01:<encrypted password>:0:7:28:7:::
test02:<encrypted password>:0:7:28:7:::
```

The defaults for new user accounts can be changed in `/etc/login.defs`:

```
PASS_MAX_DAYS   28
PASS_MIN_DAYS   7
PASS_MIN_LEN    8
PASS_WARN_AGE   7
```

Here we also see the **PASS_MIN_LEN** option, which has been increased to 8 as per the Solaris example.

To implement the equivalent of `HISTORY=5` on Solaris, PAM (Pluggable Authentication Module) configuration must take place.

Edit `/etc/pam.d/system-auth-ac` and update the following line:

```
password    sufficient    pam_unix.so sha512 shadow nullok
            try_first_pass use_authtok
```

Append `remember=5` to the line:

```
password    sufficient    pam_unix.so sha512 shadow nullok
            try_first_pass use_authtok remember=5
```

Now, if a user attempts to reuse a recent password, they'll see a message such as the following:

```
$ passwd
Changing password for user toki.
Changing password for toki.
(c)urrent UNIX password:
New password:
Retype new password:
Password has been already used. Choose another.
passwd: Authentication token manipulation error
```

This will create the file `/etc/security/opasswd` to track the password history of all system users.

Further configuration akin to `/etc/default/passwd` on Solaris can be configured via `pam_cracklib`.

```
# fgrep pam_cracklib /etc/pam.d/system-auth-ac
password    requisite     pam_cracklib.so try_first_pass
            retry=3 type= 
# man 8 pam_cracklib
```

You can read more via the section 8 manual page on `pam_cracklib`.

**Logging**

Linux does a very good job at logging via the `authpriv` syslog facility – for example, there will be messages in `/var/log/secure` such as:

```
Dec  4 16:13:39 centosb groupadd[8177]: group added to / 
            etc/group: name=testgrp, GID=1010
Dec  4 16:13:39 centosb groupadd[8177]: group added to / 
            etc/gshadow: name=testgrp
Dec  4 16:13:39 centosb groupadd[8177]: new group: 
            name=testgrp, GID=1010
Dec  4 16:13:55 centosb useradd[8183]: new user: 
            name=test01, UID=1010, GID=1010, home=/h 
            ome/test01, shell=/bin/bash
Dec  4 16:14:05 centosb useradd[8189]: new user: 
            name=test02, UID=1011, GID=1010, home=/h 
            ome/test02, shell=/bin/bash
Dec  4 16:14:42 centosb passwd: pam_ 
            unix(passwd:chauthtok): password changed for test01
Dec  4 16:14:49 centosb passwd: pam_ 
            unix(passwd:chauthtok): password changed for test02
Dec  4 16:17:50 centosb chage[8255]: changed password 
            expiry for test01
Dec  4 16:17:51 centosb chage[8258]: changed password 
            expiry for test02
Dec  4 16:30:38 centosb chage[8282]: changed password
```

Now, if a user attempts to reuse a recent password, they'll see a message such as the following:

```
$ passwd
Changing password for user toki.
Changing password for toki.
(c)urrent UNIX password:
New password:
Retype new password:
Password has been already used. Choose another.
passwd: Authentication token manipulation error
```

This will create the file `/etc/security/opasswd` to track the password history of all system users.

Further configuration akin to `/etc/default/passwd` on Solaris can be configured via `pam_cracklib`.

```
# fgrep pam_cracklib /etc/pam.d/system-auth-ac
password    requisite     pam_cracklib.so try_first_pass
            retry=3 type= 
# man 8 pam_cracklib
```

You can read more via the section 8 manual page on `pam_cracklib`.

**Logging**

Linux does a very good job at logging via the `authpriv` syslog facility – for example, there will be messages in `/var/log/secure` such as:

```
Dec  4 16:13:39 centosb groupadd[8177]: group added to / 
            etc/group: name=testgrp, GID=1010
Dec  4 16:13:39 centosb groupadd[8177]: group added to / 
            etc/gshadow: name=testgrp
Dec  4 16:13:39 centosb groupadd[8177]: new group: 
            name=testgrp, GID=1010
Dec  4 16:13:55 centosb useradd[8183]: new user: 
            name=test01, UID=1010, GID=1010, home=/h 
            ome/test01, shell=/bin/bash
Dec  4 16:14:05 centosb useradd[8189]: new user: 
            name=test02, UID=1011, GID=1010, home=/h 
            ome/test02, shell=/bin/bash
Dec  4 16:14:42 centosb passwd: pam_ 
            unix(passwd:chauthtok): password changed for test01
Dec  4 16:14:49 centosb passwd: pam_ 
            unix(passwd:chauthtok): password changed for test02
Dec  4 16:17:50 centosb chage[8255]: changed password 
            expiry for test01
Dec  4 16:17:51 centosb chage[8258]: changed password 
            expiry for test02
Dec  4 16:30:38 centosb chage[8282]: changed password
```
expiry for test01
Dec 4 16:30:39 centosb chage[8285]: changed password
expiry for test02

This makes for an excellent audit trail (as long as logs are regularly rotated and backed up, of course).

The Solaris audit tool does not currently log these actions (even via the audit daemon, but it will log password changes), but simple wrapper scripts could be written around the existing tools to perform appropriate logging.

Other Tools
Solaris provides the logins command which displays a variety of information on currently configured user accounts. An example:

```
# logins -xo
root:0:root:0:Super-User:/root:/usr/bin/
   bash:PS:112413:-1:-1:-1
daemon:1:other:1::/usr/sbin/sh:NL:082587:-1:-1:-1
bin:2:bin:2::/usr/bin:/usr/sbin/sh:NL:082587:-1:-1:-1
sys:3:sys:3::/usr/sbin/sh:NL:082587:-1:-1:-1
adm:4:adm:4::/usr/sbin/sh:NL:082587:-1:-1:-1
uucp:5:uucp:5::/usr/lib/uucp:/usr/sbin/sh:NL:082587:-1:-1:-1
nuucp:9:nuucp:9::/var/spool/uucppublic:/usr/lib/uucp/sh:NL:082587:-1:-1:-1
dladm:15:netadm:65::/usr/sbin/dladm/sh:LK:000000:-1:-1:-1
netadm:16:netadm:65::/usr/sbin/dladm/sh:LK:000000:-1:-1:-1
```

More information (including password expiration fields) can be added with `-a`. Other reports can also be generated, such as logins with duplicate UIDs (the `-d` option).

Another very useful option is the `-m` option which acts like the groups command and displays all groups that the user (specified with the `-l` option) is a member of:

```
# logins -m -l root
root            0       root            0       Super-User
other           1
bin             2
sys             3
dadm            4
uucp            5
mail            6
tty             7
lp              8
nuucp           9
daemon          12
```

Security Considerations
User access should also be limited with regards to system access and service availability. For example, we can limit access to cron and at by using `/etc/{cron,at}. {allow,deny}` files. By default, all users have access to cron and at. This may or may not be what you want. From a security standpoint, all users should be denied access by default. Therefore, have an empty `/etc/cron.deny` and `/etc/at.deny`, and a single line in `/etc/cron.allow` and `/etc/at.allow` of root. Then, grant access as necessary by appending usernames to the files. For example, if you enable system profiling via `sa1` and `sa2` on Solaris, you’ll need to add `sys` to the `-d` files.

Remote SSH access to the system can be managed via the `AllowUsers` parameter, amongst others. From `sshd_config(5)`:

```
AllowUsers
This keyword can be followed by a list of
user name patterns, separated by spaces. If specified, login is
allow only for user names
that match one of the patterns. Only user
names are valid; a numerical user ID is not recognized. By default,
login is allowed for
all users. If the pattern takes the form
USER@HOST then USER and
HOST are separately checked, restricting
logins to particular users
from particular hosts. The allow/deny
directives are processed in
the following order: DenyUsers, AllowUsers,
DenyGroups, and finally
AllowGroups.
```

/etc/security/access.conf is another access control mechanism on RHEL-derivatives and other distributions. An excerpt from the well-commented example:

```
# User "john" should get access from ipv6 net/mask
#+ : john : 2001:4ca0:0:101::/64
#
# All other users should be denied to get access from all
# sources.
#- : ALL : ALL

Simple <flag>:<username or groupname>:<ip_address>
ACLs can be defined in this way, although it could become cumbersome to manage for large sites. Remember that you'll need to ensure that `pam_access.so` is
required in any files under /etc/pam.d as appropriate if you do decide to use pam_access.

On CentOS, we can also configure /etc/security/limits.conf, and limit the following resources to specific users or groups:

```
#    – core - limits the core file size (KB)
#    – data - max data size (KB)
#    – fsize - maximum filesize (KB)
#    – memlock - max locked-in-memory address space (KB)
#    – nfile - max number of open files
#    – rss - max resident set size (KB)
#    – stack - max stack size (KB)
#    – cpu - max CPU time (MIN)
#    – nproc - max number of processes
#    – as - address space limit (KB)
#    – maxlogins - max number of logins for this user
#    – maxsyslogins - max number of logins on the system
#    – priority - the priority to run user process with
#    – locks - max number of file locks the user can hold
#    – sigpending - max number of pending signals
#    – msgqueue - max memory used by POSIX message queues (bytes)
#    – nice - max nice priority allowed to raise to values: [-20, 19]
#    – rtprio - max realtime priority
```

The format of entries in this file are:

```
<domain> <type> <item> <value>
```

As test01, we can see the new restrictions in place:

```
[test01@centoshost ~]$ ulimit -Su
20
[test01@centoshost ~]$ ulimit -Hu
30
```

pam_tally2.so can be used to keep a tally of failed logins per-account, and deny access once the number reaches a certain value. Add the following to /etc/pam.d/login and /etc/pam.d/sshd

```
auth required pam_tally2.so deny=4 unlock_time=1200
```

This would deny access to an account after 4 bad attempts, but will allow access again after 1200 seconds. The tally information is written to /var/log/tallylog. The pam_tally2 command is used to administer the tallied accounts. For example, suppose user toki was denied access. First, check the tally:

```
# pam_tally2 --user toki
Login Failures Latest failure From
toki 7 12/04/13 18:25:36 localhost
```

Once verified that these failures are not caused by malicious means, unlock the account by resetting the tally:

```
# pam_tally2 --user toki --reset
Login Failures Latest failure From
toki 0 12/04/13 18:25:36 localhost
```

pam_faillock.so also provides similar functionality.

**Privileges**

Rather than giving everyone the root password, which is obviously an extremely poor security practice, if elevated privileges are required, sudo should be configured. Sudo is installed by default on both CentOS 6.4 and Solaris 11.1 and provides a mechanism to define ACLs as to which users and groups can perform privileged actions on a server, with or without passwords. Again, the default file is very well commented, for example from /etc/sudoers:

```
## Allows people in group wheel to run all commands
# %wheel ALL=(ALL) ALL
```

**Privileges**

Rather than giving everyone the root password, which is obviously an extremely poor security practice, if elevated privileges are required, sudo should be configured. Sudo is installed by default on both CentOS 6.4 and Solaris 11.1 and provides a mechanism to define ACLs as to which users and groups can perform privileged actions on a server, with or without passwords. Again, the default file is very well commented, for example from /etc/sudoers:

```
## Allows people in group wheel to run all commands
# %wheel ALL=(ALL) ALL
```
# Same thing without a password
# %wheel ALL=(ALL) NOPASSWD: ALL

## Allows members of the users group to mount and unmount the 
## cdrom as root
# %users  ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/ cdrom

## Allows members of the users group to shutdown this system
# %users  localhost=/sbin/shutdown -h now

You should use the `visudo` command to edit the `/etc/sudoers` file, as it performs sanity checks before saving the file and possibly corrupting the live `sudoers` file if there are errors in your syntax.

If there is an insistence on sharing a root password, then the number of people knowing that password should be limited. All access to the root account should be made via `su` and not via direct root login, which should be limited to the system console only. This will provide logging and an audit trail.

**Conclusion**

This article has described some of the major aspects of user, group and password management on the CentOS 6.4 and Solaris 11.1 Operating Systems. User account management is a complex subject, so only the core aspects have been covered. The manual pages and system documentation should be consulted for further information.

---

**ABOUT THE AUTHOR**

*Toki Winter is an experienced UNIX/Linux System Administrator with over a decade of experience managing large enterprises and running many thousands of services. His website, http://www.tokiwinter.com, contains many useful articles, HOWTOs and tips for the advanced UNIX administrator.*

wwwbsdmag.org
How did you get your start in the information security field?

Peter N.M. Hansteen: I’ll risk sounding a little blunt here, and say that it was really a matter of a series of accidents that led to, well, a known result. Early on I had what you might call a rather meandering career path before I finally started pointing myself in a generally IT-ish direction. Fortunately while I was taking night classes in IT subjects and working a day job in a very junior clerical position at the Norwegian School of Economics here in Bergen, I got an early introduction to the Internet as it was then in the mid to late 1980s. I remember distinctly that a fair number of the machines we encountered at the other side of telnet, archie, ftp and other services ran something slightly exotic called BSD Unix.

A few job changes later and I found myself in a position where I was the person in charge of information security and everything IT-ish for myself and about a dozen colleagues. As the inevitable Internet commercialization came around I had a slight edge after some early exposure and hanging around BBSes in the meantime. But then again we had some wonderful security failures too, as far as I can tell not too dangerous and never really breaking anything important, but well, the stories are out there in some form if you poke around USENET archives. Enterprising readers will know where to look.
What drove you to pursue information security?

PNMH: Information security, again, is part of the bigger picture. You want to provide a convenient working environment as well as making sure you keep your colleagues safe from harm, all the while doing your best to implement a regime that protects whatever the organization’s assets are. For my own part it all grew out of that motivation. The process was quite gradual. And of course gradually you build up a toolchest. There are invariably applications or entire environments that you would dearly like to take out of the equation that also happens to be something your client can not be moved to do without. For my own part I ended up with a preference for open source tools in general and OpenBSD in particular. That position evolved in part from various less pleasant experiences with the various proprietary systems, and partly from the rather obvious insight that with open source tools, you actually can check what the tools do and change or enhance any part of the toolchain if you want to.

Then again, whatever you do and how ever you choose to run your security efforts, the security bits have to be integrated into your environment. Basically the tools and procedures need to be part of the normal, ordinary way of going about your business. If your strictly enforced security regime with tools and procedures gets in the way of how the organization needs to run its business, your users will find ways to subvert your goals and you may find yourself exposed. It’s the you made the thing foolproof, so they went ahead and created a bigger fool problem coming back at you.

What do you see as the biggest challenge to information security five years down the road?

PNMH: Well, to start with there are four things we can be absolutely sure will be as problematic five years from now as they are today: Bad design decisions, needlessly growing complexity, implementation bugs, and your trusted users’ actions, including your own. For the first three the constant, ongoing code audit of the type the OpenBSD project practices and preaches will give you a head start. But apart from stating the obvious, there are a few other worrying developments that have been happening for a while and have only recently started to come to the general public’s attention.

One such development is the growing tendency of governments, even Western ones, to demand the right to peek ever more closely into people’s private information, with little or no accountability. The European Union’s Data Retention Directive is one such piece of legislation, which mandates that any traffic logs you may be generating for your own needs have to be kept around for longer than any sane techie would think of, just in case law enforcement wants to take a peek. You could of course say that the original intentions were good and point to the so-called war on terror. But we have already seen the motivation morph into the need to catch child molesters, then it got tweaked a little more to be included as a weapon in the decades-old war on drugs and recently it’s been found to be vital in the struggle to catch traffic offenders, beaten to the punch only by a very misguided chunk of the media publishing industry, which for good measure seems to be intent on running its own little branch of law enforcement in their very own style.

The same ugly picture includes various national laws that codify warrantless wiretapping and other forms of fine grained surveillance, and there is even legislation on the way that mandates various forms of censorship that may lead to serious technical issues in the name of copyright enforcement. All taken together it looks like a fairly thorny path ahead, and it’s worth keeping in mind that all of those things that sound scary enough for individuals pose a real risk for companies too. To some extent we’ve always had industrial espionage, but to West Europeans at least the idea that your own government could realistically be the ones trying to pry into your confidential information is somewhat new and quite unpleasant.

At the end of the day, bugs of any kind and social engineering will return to bite us, and we won’t be rid of either any time soon. Our adversaries will continue to rely on those techniques. Well designed tools and good code, validated and audited in full public view will help, as will educating your users. Keep in mind too that in this context you, the security professional, are very much a user yourself, with access to elevated privileges that may mean when you do screw up, the situation could escalate into something far more dangerous than run of the mill user’s goofs.

Does the OpenBSD version numbering approach confuse people?

PNMH: I suppose it does confuse people that in OpenBSD, the version number is just another identifier, and it gets incremented by exactly 0.1 every six months.

The reason OpenBSD does it that way is that the project has chosen to live by a strict six month development cycle. The development cycle is itself split into roughly four months of introducing new or improved features followed by two months of stabilization leading up to cutting a release and sending it off to production at some never-preannounced date. For the development team this means that large reworks of code will have to be split into chunks that will realistically fit within that timeframe.
The much-ballyhooed and very useful syntax changes that appeared in PF over the OpenBSD 4.6 and 4.7 releases had in fact been works in progress for some years when they hit the tree for general use. For last November’s release, 5.0 just happened to be the next increment in line. The release did have some major new features, for PF the prio keyword is the first part of a new traffic shaping engine that will eventually replace the venerable ALTQ when the time comes.

There is kind of a roadmap in place, but the developers have not officially committed to a timetable or specific release when ALTQ is supposed to be replaced. It will happen when the new code is ready and clearly better than the older one. When something new and exciting is committed, I hope to be one of the first to write a blog post about it. My PF tutorials tend to include at least some mention of recent developments, too.

**Do you believe all the regulations set forth regarding information security have helped or hindered information security growth?**

**PNMH:** First of all, there is more legislation that’s relevant to information security today than there was earlier, and security professionals need to be aware of what rules apply to them. Some legislation may have been beneficial, if for example it was needed in order to codify clear standards of ethical conduct. Basically you need a working knowledge of what rules apply. So the various rules and regulations have made life anything from slightly more complicated to somewhat painful in recent years, depending on where you are and what you do.

If you work in several jurisdictions, you may need to get a lawyer or even a judge to affirm which set of rules apply in each case, and if the precedence of rules is unclear or worse the rules are even slightly incompatible or unclear, your legal fees could become substantial.

Again it’s important to be aware that recent legislation in the US and elsewhere written with the intention of short-circuiting the normal due process rules in certain types of criminal cases, notably those labeled ‘terrorist’ by the prosecution. Unless those rules are found unconstitutional in a hurry, we should expect to see information security professionals behind bars for indefinite periods soon enough.

**Is there a better way to allow root access for remote admins?**

**PNMH:** Heh. There has been a lot of discussion on just what level of immediate access is appropriate for admins when they are in a hurry, but realistically the question boils down to this: What level of exposure to the various threats, including the risk of your own mistakes, is appropriate in your context?

I don’t believe there is an easy one size fits all option available. Your analysis of the specific context, with its own set of risks and probabilities and anticipated threat factors dictates what is appropriate.

But reeling back a bit, your question is really about the basic conflict or tradeoff that admins see between convenience on the one hand and security on the other when they need to access critical devices. It’s so very convenient to go directly to the maximally permissive settings so you can do anything you like without getting caught up in red tape.

When it comes to what constitutes acceptable risk, it really is up to you. If you, after appropriate risk analysis, are confident that logging in to a remote device with the highest possible privilege is appropriate, if you are equally confident that you can effortlessly recover from any mistakes you make while running with maximum privilege and you consider the risk that anyone not formally authorized to reach that level will manage to do so is negligible to non-existent, you are at liberty to go directly to root.

I tend to advocate disallowing direct login to any privileged account, to encourage use of encryption of the strongest practical kind and when appropriate and available, key based authentication or some sort of two factor authentication system. Mainly because I know that I am not infallible, and in some contexts I need a reasonable assurance that anyone attempting unauthorized access would need to expend enough effort that my systems would detect the attempt.

I dislike running with elevated privileges whenever it isn’t strictly necessary, mainly because I know that I’m human and will make mistakes, and that configurations can break in unexpected ways. There are, for example, failure modes on some Unix-ish systems that would land you with / as your home directory and no warning that’s where you are other than – if you’re lucky – a command line prompt that looks subtly different from what you are used to seeing. In those contexts, it’s essential to do the right things, and your confidence that you will have *grace under pressure* will be sorely tested.

A large part of the problem is to ensure that any task in the system runs with an appropriate level of privilege. In the OpenBSD project, a lot of work has gone into properly implementing privilege separation in the various daemons. In effect, making sure only those parts of the system that need elevated privilege ever achieve that privilege, and in most cases the program gives up the privilege once the task such as binding to a port below 1024 has been achieved. The most immediately user-visible consequence...
is that you will find the OpenBSD password database pre-populated with a number of special-purpose users (most of them with names that start with an underscore character ‘_’), defined specifically to run services at their appropriate privilege levels.

The privilege separated OpenBSD system is out there and available for daily use, and I would encourage your readers to try it out. There are interesting efforts going on in other projects as well, with the main keywords being RBAC or Role Based Access Control – essentially a deconstruction of the user authentication and authorization (implemented among other places in the most recent Solaris releases), and from the opposite end of the table, fine grained capabilities models for process privilege separation, with the FreeBSD project’s Capsicum project (if I understand correctly to hit mainstream in FreeBSD 9) on my short list of things to look into in the near future.

But the increased complexity that grows naturally from these approaches also means the code and configuration needed to fit the code to your purposes is harder to do correctly, and so we are almost certainly entering dangerous territory for that reason alone. It will take significant development effort to rein in those concepts into something manageable for the average sysadmin, assuming we’re also able to squash enough bugs in the process to make the effort worthwhile.

What new concepts or applications are available, or coming available soon, for firewalls?

PNMH: The firewalls concept in its simplest form – and that’s what people get hung up on – is rather simplistic. The main decision is to block or pass. Modern firewalls do a lot more of course, including but not restricted to failover and redundancy with CARP and pfsync or VRRP, network address translation and even IPv4 to IPv6 conversion, redirections, load balancing and traffic shaping. The good ones even adapt to network conditions via adaptive state timeouts or can be configured with state tracking tricks that fend off excessive traffic of specific kinds.

And of course there is more, but there is a tendency for news about interesting technical development to drown in marketing hype, so I may be ignoring important work that’s going on out there. Personally I think the authpf system – OpenBSD’s and PF’s non-interactive shell that loads rules on a per user basis – is one type of feature that I think will see a lot more attention and wider use in the future. It’s so obviously a good thing to tie what the network lets you do to your user or group identity or to a set of role based criteria.

Come to think of it, most of these advanced firewall features are seriously under-used and not as well understood in the community at large as we would have liked. But perhaps the identity or role centric setups are the ones with the most scope for interesting development over the next few years, if the added complexity can be managed somehow.

How does BSD pf compare to iptables, ipfw, ipfilter or other firewalls? What is its strength or weakness?

PNMH: The short answer, coming as it would from the author of The Book of PF, is obviously that the other ones suck. But seriously, since I kind of abandoned the other ones in favor of PF at some point, I think it’s better to at least start answering the question by describing some of the features that attracted me to PF over the other ones. Then we’ll get around to any weak points if we can still remember them after a while.

It’s important to remember that PF is developed as an integrated part of OpenBSD, and one of the important design goals has always been that it should be very usable for OpenBSD users. This means that all the features I’ve touched on earlier are within easy reach directly from your pf.conf configuration file or somewhere equally accessible.

One usability feature I appreciate a lot is called atomic ruleset load. It’s perhaps easier to explain why this is important if we look at the other ones: iptables and ipfw configurations are actually shell scripts, where each rule is loaded as a separate command. This means that if you press [Ctrl-C] while the script is executing, you have very little control over what rules are actually enabled. More likely than not, some lines of your script were never executed, meaning that your configuration did not load completely, with unpredictable results. IPfilter’s developer apparently did not trust the software to keep track of loaded rules by itself and recommended flushing previous rules before loading a new configuration.

None of this is necessary with PF – if your rule set is tactically valid, it will load, completely replacing the previous one. There is no need to flush existing rules, unless you want to make sure you have a a period of ‘pass all’ to give miscreants a break until you load the next valid rule set, and running a real risk of disrupting valid traffic (think timeouts due to disappearing redirections) that would have seen no trouble on a clean ruleset load.

If you think this means that PF configurations are totally static, you’re wrong. If you need to adjust the contents of your rule set on the fly, your best bet is to create what PF calls an anchor – essentially a named sub-ruleset, and yes, you can have several – where you or applications you write can insert and manipulate rules dynamically, something Apple appears to have used to great effect in their port to MacOS. Apple even wrote some enhancements to the anchor
loading code, but unfortunately they wrapped their new bits in #ifdefs with a separate license, so the extended functionality will not easily make it back into the mainstream PF code. You can look up my Call for testing article (see the references at the end) for more details.

And of course for simpler operations like singling out hosts that need special treatment, you can manipulate tables of IP addresses even outside anchors, or you can use state tracking options magic to move IP addresses into tables, and use the tables in your filtering criteria.

From my experience, PF and related tools on OpenBSD provide you with the sanest working environment available for interacting with the TCP/IP stack so you can make your equipment perform the way it’s supposed to. None of the other tools come even close, in my opinion, in either admin friendliness or performance.

Will the next intrusion platform be mobile devices?

PNMH: To some extent, or possibly even to a large extent, I think the shift has already happened, in the sense that the focus of would-be intruders is changing more or less in step with the mainstream user and the perceived high-value targets. I’m not suggesting that the installed base of PCs will be going away anytime soon, most of those are well past their use by date anyway, but rather that the Windows PCs that today still make up the largest part of the installed base are destined to become less important over time if current trends continue more or less as we see them today.

Mobile devices are getting a lot of attention these days, and malware targeted at them is of course getting some too. The situation for mobile devices designers today is somewhat parallel to the situation when PCs were introduced to the Internet, but there are important differences.

One such difference between back then and now is that a large part of the PC related business is still aimed squarely at patching or working around security bugs in the most common desktop operating environment. That, and the fact that there are more network-savvy developers out there today than at any time earlier makes me a little hopeful that at least some of the grosser mistakes of PC networking history will not be repeated by mobile device developers. Also, so far we have avoided the monoculture that helped make PCs on the Internet such easy targets. I’m not suggesting that the installed base of PCs will be going away anytime soon, most of those are well past their use by date anyway, but rather that the Windows PCs that today still make up the largest part of the installed base are destined to become less important over time if current trends continue more or less as we see them today.

Mobile devices are getting a lot of attention these days, and malware targeted at them is of course getting some too. The situation for mobile devices designers today is somewhat parallel to the situation when PCs were introduced to the Internet, but there are important differences.

One such difference between back then and now is that a large part of the PC related business is still aimed squarely at patching or working around security bugs in the most common desktop operating environment. That, and the fact that there are more network-savvy developers out there today than at any time earlier makes me a little hopeful that at least some of the grosser mistakes of PC networking history will not be repeated by mobile device developers. Also, so far we have avoided the monoculture that helped make PCs on the Internet such easy marks. Mobile devices vendors have a real choice in software stacks, and at least the two dominant operating environments in the smartphone space (Apple and Android) are both vaguely Unix-based and use open source components to some degree, which seems to me like their designers are capable of making intelligent decisions.

That said, I’m fairly sure that even in those environments, users and miscreants will find ways to exploit bugs, and some subset of users will always be willing to do things that are simply not smart things to do. One example comes to mind – users of Apple phones decided that their phones ran a system that was unix-like enough that it should be able to accommodate a Secure Shell (ssh) server, and somebody managed to port the software. Only that developer decided to provide a setup with a default password, and there were several reports of phones that were taken over via ssh, thanks to the known default password that the user never bothered to change.

Now I’m geek enough to appreciate the attraction of having a shell login to the phone you carry in your pocket, but (as I noted in a slashdotted blog post at the time) the point here is not that sshd is an insecure piece of software. It isn’t. The lack of security comes from not bothering to change your password from a well-known default value.

Something similar is bound to happen again, where a user makes a stupid mistake that has security implications. If we’re lucky the damage will be limited to the user’s own equipment, but if that user is also a developer and by design or accident inserts exploitable code in other users’ mobile devices, the damage could become more widespread.

It’s also worth keeping in mind that even if mobile devices seem relatively boring by modern PC standards and may or may not contain useful data, they may still be useful to botnet herders. A typical smartphone today has general processing power at least on par with a run of the mill PC at the time the network dependent malware started turning up on the Microsoft platform, and it’s almost certainly on a better network connection than most PCs were back then. If your smartphone doubles as your wallet, all the more reason to pay attention.

How can these mobile devices be protected?

PNMH: If the mobile devices industry indeed manages to avoid making the same mistakes as the PC industry before it, I think we have something of a head start. Over the years what passes for IT security has focused on enumerating badness (do read Marcus Ranum’s essay linked to in the references for more on that) and in the process diverting attention from the root cause issue that a certain software marketer was, for quite some time, reluctant to even acknowledge that there were bugs to be found in their software.

It may not have been obvious at the time Marcus was writing that essay, but history has taught us that the approach the PC industry took to security at the time – heaping another level of complexity on top of buggy software in the name of security – is not necessarily an improvement in real terms, even if the new layer somehow provides a
workaround for the nasties you know about in the original code. The added complexity most likely means that your debugging gets harder for the next round of problems.

In a way it would be nice if mobile device designers started basing their systems on OpenBSD, which is probably the general purpose system that has been developed and maintained with the most attention to security. I want a phone I can trust, and preferably one that’s open enough for qualified developers to hack on. And the same applies to tablets and other devices too, of course.

Regardless of what technology the devices are based on, I think a combination of user education and operators paying attention to end user equipment is the way forward. If operators are able to take some of the system administration workload off their end users’ hands for a nominal fee, it could turn into a profit center.

It really boils down to a sane system administration regime – don’t run any services that are not required for your use case, log properly and pay attention to what your logs say, update your systems at intervals and definitely when security relevant bugs have been fixed.

On the other hand, in addition to user education and the offer of handholding we may need a measure of negative reinforcement – one approach is to mimic the way we treat pets or livestock and their owners. Dogs and computers both are capable of autonomous actions to some extent, so it might be a useful parallel. Dog owners are used to cleaning up the messes their pets make on sidewalks, and if the animal bites somebody, the owner is usually responsible for paying for the damage. Sufficiently stupid behavior with regard to your pet can sometimes earn you a charge of reckless endangerment. I think you can validly argue that a similar regime should apply to owners of fairly damage-capable computing devices.

**How can these mobile devices be firewalled?**

**PNMH:** On a technical level, I think that problem is very close to being solved. The existing tools could be adapted fairly easily to fit a roving user scenario (some people are already paying attention), and some of the anticipated developments I mentioned earlier may make the devices even easier to use. But once again, operators and service providers could play a significant role if they manage to come up with useful ways to interact with users’ devices. And of course we need to stomp out the snake oil salesmen, if we can’t scare them off right away by building sanely constructed devices with trustworthy software.

**How do you even know if someone is attempting to access your mobile device or using it to run ssh login attempts against remote systems?**

**PNMH:** On the current crop of devices, I think you’d be blissfully ignorant of any such attempts until either your phone starts doing something unexpected or your next bill turns up with a lot more traffic to pay for than you had expected.

With any of the devices that are vaguely unix-based it shouldn’t be very hard to log properly, and once again I think operators should be looking seriously into offering their users some kind of log monitoring and other admin services in order to help run mobile devices sanely. Intelligently designed mobile device management services could become the real differentiator in the telecom operator market. I hope the operators are paying attention.

And finally, for penetration testers out there, there will always be bugs out there to hunt for and exploit, and if you have a hard time finding those, you can always go for layer 8 or 9 techniques :)

Happy hacking!

---

**References**

The references are listed in roughly the order they’re mentioned in the text, read them for further treatment of some of the issues I mentioned here.

- The OpenBSD project [http://www.openbsd.org/](http://www.openbsd.org/)
- The FreeBSD project [http://www.freebsd.org/](http://www.freebsd.org/)
- PF tutorial home page [http://home.nuug.no/~peter/pf/](http://home.nuug.no/~peter/pf/)
- The Capsicum Project at Cambridge University, [http://www.cl.cam.ac.uk/research/security/capsicum/](http://www.cl.cam.ac.uk/research/security/capsicum/)
- The Book of PF (second edition) [http://nostarch.com/pf2.htm](http://nostarch.com/pf2.htm) or from good bookstores everywhere
With the Collapse of Red Flag Software (the World’s Second-largest Linux Distributor) is the Dream of Linux on the Desktop Even Further out of Reach?

When Red Flag Linux was launched in 2007, there was much fanfare in the Open Source community. The most populated country in the world had embraced the vision, backed by government, and even a grudging olive branch had been thrown towards Microsoft in developing a Windows XP like interface. What could possibly go wrong? At time of writing it is unclear exactly why the project collapsed so spectacularly; some cite the Chinese Academy of Sciences' removal of funding due to the competition from Red Hat and SUSE Linux. What is clear however, is that another large government backed computer project has fallen by the wayside.

This is a sad day for the Open Source movement. While the cynical amongst us may suspect back doors and all sorts of underhanded compromises that go hand in hand with government surveillance and not shed a tear over the demise of Red Flag from an ethical perspective, the fact remains that domination of the world’s largest marketplace is back in the hands of commercial interests, other than the enlightened few stalwarts that decide to download their own software – that is, of course, if it is available via government controlled firewalls or via a DVD from a friend. Under the circumstances it would be hard to accept a score other than Communism 0 – Capitalism 1.

Anyone with a scintilla of commercial reality understands that the desktop is dominated by Microsoft, as the majority of corporations have adopted MSC solutions. This popularity has spread across to the consumer marketplace but with one exception – mobile and tablet devices. In the early age of the motor car, the market was dominated by Ford with their innovative vision of mass production. Everyone else then followed suit and, to this day, very few independent motor manufacturers remain. But where does Ford rank today? Well behind General Motors, Volkswagen and Toyota in terms of production.

So the cyclical curse of the capitalist marketplace once again claims another scalp – the innovator, the creative – once on top doesn’t always finish first. So maybe MSC isn’t in such a strong position after all.

MSC is at a critical juncture in its history. There is a serious move away from the classic in-house desktop / server relationship with the success of tablets and mobile phones. Organisations are thinking more and more along the lines of remote desktops, virtualisation and bring your own device. Maybe the question isn’t what Operating System will dominate the desktop, but what method will be used to deliver applications? If the move towards thin-client takes off, MSC will need to morph away from its traditional model for the corporates – Servers, Desktops, Applications, and Developer tools.

There is another issue at stake here, apart from the ethical issues of moving applications and data off to some server farm somewhere. Windows 8, like Ubuntu Unity, has caused consternation amongst the old school by radically re-designing the user interface in an attempt to bring cohesion across devices. It has been a Marmite moment – you either love it or hate it. At the moment, the corporates hate it, and those who are not committed to the change from a Start button or classic menu anchored to the top or bottom of the screen struggle. I recently had to explain to a friend, who had purchased a new consumer-grade laptop with Windows 8 installed, that he was basically stuck with it unless he forked out for a Windows 7 licence – a discussion that involved much swearing and slapping of the forehead. Friends come and go, but enemies accumulate and MSC is building a dedicated following of the latter with its short-sighted “Our way or the highway” mentality.

Microsoft’s nemesis on the other hand has it all wrapped up as far as the user interface is concerned. The humble hyper-link is clicked on 1 x 10X per day where X is greater
With the Collapse of Red Flag Software...

than 10. When it comes to vox populi, or the voice of the people, any computer interface is going to fail on the basis of these statistics. No focus group, design team or engineer on the planet can overcome the intimacy that billions of people have developed with clicking on an HTML link. I would guess that Bob Bemer didn’t have a focus group on hand to test efficacy. So Google, the future is yours in terms of statistical dominance.

Yet we still have the problem of the UI bling factor – the pretty, touchy feely effect that Apple has embraced and made almost a religion out of. While the hyper-link is cold and efficient, exploding windows, cute animal sounds and great font rendering bring élan to an emotionally sterile environment. MSC has never quite penetrated this US West Coast paradox, yet we see the same trends with those who love their Android O/S and the touch screen. It is the “wow” factor.

At the end of the day, software interfaces need to be just that – an interface. The same rules apply to the design of a car, a cheese grater or a garlic crusher. Some will be utilitarian, some revolutionary. What goes on underneath the bonnet will be hidden to the majority of users, but first impressions really matter. We all intuitively understand good design – it has that feel about it, an aura, a quality you just cannot put into words. It pulls you into itself, re-enforcing your understanding of the universe yet at the same time challenging you to explore further. It is greater than the sum of its parts. So maybe the demise of Red Flag Software is a mercy killing rather than an assassination.

If the move to the cloud and thin client is the next revolution, it matters little what that thin client will be, as the forces of mass adoption will dictate how people interact in cyberspace. The O/S will become less and less important, and the user experience and interface will become more so. And that is where the trojan horse of Open Source will dominate – the power behind the throne.

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.
Dr. Web 9.0 for Windows — the rapid response anti-virus

1. Reliable protection against the threats of tomorrow
2. Reliable protection against data loss
3. Secure communication, data transfer and Internet search

www.drweb.com
Free 30-day trial: https://download.drweb.com
New features in Dr. Web 9.0 for Windows: http://products.drweb.com/9
FREE bonus — Dr. Web Mobile Security: https://download.drweb.com/android
Great Specials
On FreeBSD® & PC-BSD® Merchandise

$39.95
FreeBSD 9.1 Jewel Case CD Set
or FreeBSD 9.1 DVD

$29.95
PC-BSD 9.1 DVD

$49.95
The PC-BSD 9.0 Users Handbook
PC-BSD 9.1 DVD

$99.95
The FreeBSD CD or DVD Bundle
Inside each CD/DVD Bundle, you'll find:
FreeBSD Handbook, 3rd Edition
Admin Guide FreeBSD 4.0 CD/DVD
FreeBSD T-Shirts

Stylish Dress Attire
Look Your Professional Best

Comfy Apparel
Stay Warm in Zip Ups & Pullovers

T-Shirts
Lots of Styles to Choose From

FreeBSD 9.1 Jewel Case CD/DVD ....................... $39.95

CD Set Contains:
Disc 1 Installation Boot LiveCD (I386)
Disc 2 Essential Packages Xorg (I386)
Disc 3 Essential Packages, GNOME2 (I386)
Disc 4 Essential Packages (I386)

FreeBSD 9.0 CD .............................................. $39.95
FreeBSD 9.0 DVD .......................................... $39.95

FreeBSD Subscriptions
Save time and $$$ by subscribing to regular updates of FreeBSD

FreeBSD Subscription, start with CD 9.1 ..................... $29.95
FreeBSD Subscription, start with DVD 9.1 ..................... $29.95
FreeBSD Subscription, start with CD 9.0 ..................... $29.95
FreeBSD Subscription, start with DVD 9.0 ..................... $29.95

PC-BSD 9.1 DVD (Isotope Edition)
PC-BSD 9.1 DVD ........................................ $29.95
PC-BSD Subscription .................................... $19.95

The FreeBSD Handbook
The FreeBSD Handbook, Volume 1 (User Guide) .............. $39.95
The FreeBSD Handbook, Volume 2 (Admin Guide) .............. $39.95

The FreeBSD Handbook Specials
The FreeBSD Handbook, Volume 2 (Both Volumes) ............ $59.95
The FreeBSD Handbook, Both Volumes & FreeBSD 9.1 ....... $79.95

PC-BSD 9.0 Users Handbook ............................... $24.95
BSD Magazine ............................................. $11.99

The FreeBSD Toolkit DVD ................................. $39.95
FreeBSD Mousepad ...................................... $10.00
FreeBSD & PCBSD Caps .................................. $20.00
BSD Daemon Horns ..................................... $2.00

For even MORE items visit our website today!
www.FreeBSDMall.com
Developing for Amazon Web Services? Attend Cloud DevCon!

June 23-25, 2014
San Francisco
Hyatt Regency Burlingame

www.CloudDevCon.net

Attend Cloud DevCon to get practical training in AWS technologies

- Develop and deploy applications to Amazon’s cloud
- Master AWS services such as Management Console, Elastic Beanstalk, OpsWorks, CloudFormation and more!
- Learn how to integrate technologies and languages to leverage the cost savings of cloud computing with the systems you already have
- Take your AWS knowledge to the next level – choose from more than 55 tutorials and classes, and put together your own custom program!
- Improve your own skills and your marketability as an AWS expert
- Discover HOW to better leverage AWS to help your organization today

Register Early and SAVE!

A BZ Media Event
CloudDevCon

Amazon Web Services and AWS are trademarks of Amazon.com, Inc.