MAGAZINE

BSD

FOR NOVICE AND ADVANCED USERS

TLS Hardening

HOW TO HARDEN TLS AGAINST THE USUAL ATTACKS

SETTING UP YOUR OWN PACKAGE CLUSTER IN MIDNIGHTBSD

SAVING TIME AND HEADACHES USING THE ROBOT FRAMEWORK FOR TESTING

GETTING TO GRIPS WITH THE GIMP – PART 5

VOL.8 NO.06
ISSUE 06/2014(59)
1898-9144

FREENAS MINI
STORAGE APPLIANCE

IT *SAVES* YOUR LIFE.

HOW IMPORTANT IS YOUR DATA?

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

NOW IMAGINE LOSING IT ALL

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

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

Example of one-bit corruption

THE SOLUTION

The FreeNAS Mini has emerged as the clear choice to save your digital life. **No other NAS in its class offers ECC (error correcting code) memory and ZFS bitrot protection to ensure data always reaches disk without corruption and *never degrades over time*.**

No other NAS combines the inherent data integrity and security of the ZFS filesystem with fast on-disk encryption. No other NAS provides comparable power and flexibility. The FreeNAS Mini is, hands-down, the best home and small office storage appliance you can buy on the market. **When it comes to saving your important data, there simply is no other solution.**

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

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

iXsystems®

http://www.iXsystems.com/mini

intel® inside™ ATOM™

FREENAS CERTIFIED STORAGE

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

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

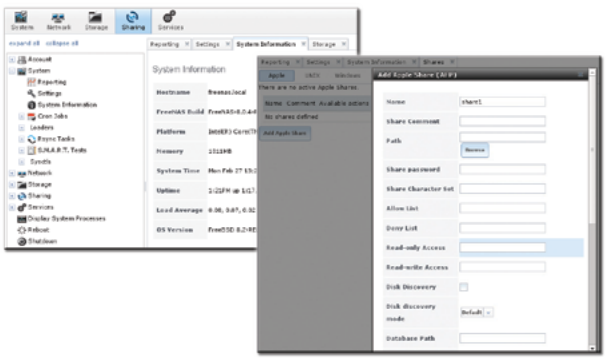
MAKE IT EASY ON YOURSELF

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

Every FreeNAS server we ship is...

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

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



FreeNAS 1U

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

FreeNAS 2U

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

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

Dear Readers,

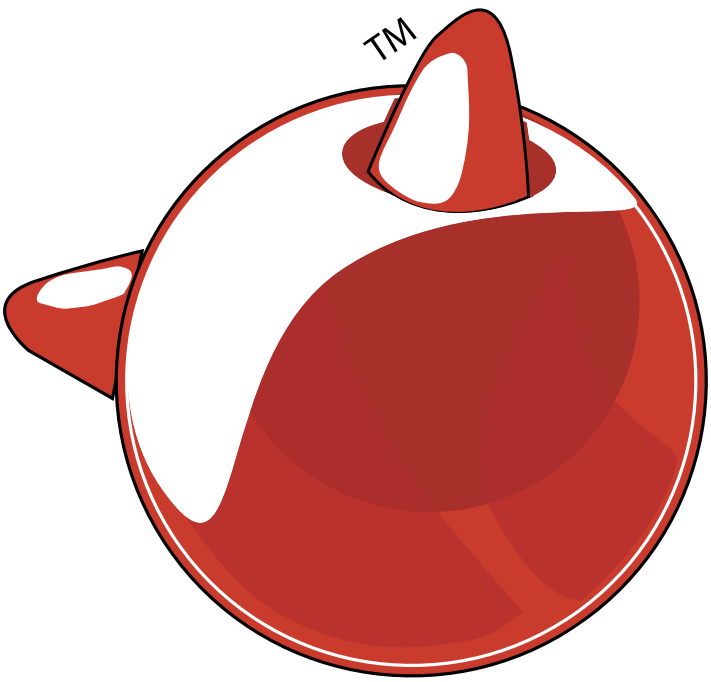
You are about to read the TLS Hardening Issue from BSD Magazine. *You have the chance to learn how to harden TLS against the usual attacks. What is more, our experts will teach you how to set up a package cluster in MidnightBSD and use the Robot Framework for testing. Finally, you may find of interest the 5th part of the GIMP tutorial provided by Rob Somerville in which he teaches you how to modify faces and hair colour.*

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

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

Hope you enjoy the issue.

Ewa & BSD Team



MAGAZINE BSD

Editor in Chief:
Ewa Dudzic
ewa.dudzic@software.com.pl

Contributing:
Michael Shirk, Andrey Vedikhin, Petr Topiarz, Charles Rapenne, Anton Borisov, Jeroen van Nieuwenhuizen, José B. Alós, Luke Marsden, Salih Khan, Arkadiusz Majewski, BEng, Toki Winter, Wesley Mouedine Assaby, Rob Somerville

Top Betatesters & Proofreaders:
Annie Zhang, Denise Ebery, Eric Geissinger, Luca Ferrari, Imad Soltani, Olaoluwa Omokanwaye, Radjis Mahangoe, Mani Kanth, Ben Milman, Mark VonFange

Special Thanks:
Annie Zhang
Denise Ebery

Art Director:
Ireneusz Pogroszewski

DTP:
Ireneusz Pogroszewski
ireneusz.pogroszewski@software.com.pl

Senior Consultant/Publisher:
Paweł Marciniak
pawel@software.com.pl

CEO:
Ewa Dudzic
ewa.dudzic@software.com.pl

Publisher:
Hakin9 Media SK
02-676 Warsaw, Poland
Postepu 17D
Poland
worldwide publishing
editors@bsdmag.org
www.bsdmag.org

Hakin9 Media SK is looking for partners from all over the world. If you are interested in cooperation with us, please contact us via e-mail: editors@bsdmag.org.

Security

06 TLS Hardening
Emmanuel Dreyfus

Emmanuel Dreyfus will present in his document TLS and how to make it secure enough, as of Spring 2014. Protocols known to be secure will be cracked and will be replaced with better versions. Fortunately, we will see that there are ways to assess the current security of your setup, but this explains why you may have to do research beyond this document to get up-to-date knowledge on TLS security.

Admin's Corner

20 Setting up Your Own Package Cluster in MidnightBSD
Lucas Holt

Magus is the package building software for MidnightBSD mports. It allows system administrators managing multiple systems to generate custom packages for their own systems. It also allows developers to test changes to mports without committing code to the subversion repository. What you will learn is the history of the Magus package software and why it was written, and how to install Magus and use Magus.

Design

22 Getting to Grips with the Gimp – Part 5
Rob Somerville

Professional graphic designers are often called to retouch images. In this tutorial, we will take this one step further and transplant the face of one model onto another. The trickiest part of this exercise is matching the skin tone and making the appearance realistic. While no humans were harmed in the creation of this tutorial, my apologies to the models concerned – you look far better in your original images! In the fifth part in our series on the Gimp we will learn about layer masks and how to modify faces and hair colour.

Testing

36 Saving Time and Headaches Using the Robot Framework for Testing
Carlos Antonio Neira Bustos

As a developer you usually start by designing your solution, then you start coding, then you start creating your test cases and turn it over to QA, and that is all. If you did it right, there is no rework needed to be done and all is well. Then your new assignment comes, and you again start designing your test cases, but this takes time. Maybe you test only the new functionality and ship it, but somewhere along the way you caused a regression because there was an unknown bug that your new code made appear, but now will you take the heat for it? If you value your time, you should start automating your testing procedures. You will save time in the long run. First, you must take the time to build your library of test cases but after you have completed it, it will save you time and headaches in the future.

Meet Our Experts

42 Interview with Siju Oommen George
BSD Team

Column

44 A recent poster on http://www.theregister.co.uk was lamenting that with all the recent security failures and the increase of patching IT is not fun any more. Are we facing a new dark period in the technology sector?
Rob Somerville

TLS Hardening

This document presents TLS and how to make it secure enough, as of Spring 2014. Of course, all the information given here will become outdated with time. Protocols known to be secure will be cracked and will be replaced with better versions. Fortunately, we will see that there are ways to assess the current security of your setup, but this explains why you may have to do research beyond this document to get up-to-date knowledge on TLS security.

What you will learn…

- What is TLS and how it is used
- How to secure a TLS installation

What you should know…

- Basic TCP/IP networking knowledge
- Some Unix system administration background

We will first introduce the TLS protocol and its underlying components: X.509 certificates, ciphers, and protocol versions. Next, we will have a look at TLS hardening for web servers, and how to plug various vulnerabilities: CRIME, BREACH, BEAST, session renegotiation, Heartbleed, and others. We will finally see how the know-how acquired on hardening web servers can be used for other protocols and tools such as Dovecot, Sendmail, SquirrelMail, RoundCube, and OpenVPN.

We assume you already maintain services that use TLS and have basic TCP/IP network knowledge. Some information will also be useful for the application developer.

An introduction to TLS

TLS stands for Transport Layer Security. It is an encryption and authentication layer that fits between the transport and application levels in the TCP/IP network stack. It was specified by IETF in 1999 as an enhancement over Netscape's Secure Socket Layer (SSL), which is why we often see the SSL term used instead of TLS.

TLS is easy to add on top of any TCP service, and this is why it has grown so popular and become available for many protocols. For instance, HTTP can be used over TLS, using the well-known https:// URL. It works the same way for SMTP(S), IMAP(S), POP(S), LDAP(S), and so on.

X.509 certificates

The main goal of TLS is enforcing confidentiality and integrity. This cannot happen if the remote peer is not properly authenticated – who cares about having a secure channel if we do not know who we are speaking to? Attacks where an intruder slips between the two legitimate parties are known as *Man in the Middle* (MITM) attacks. In such a setup, a secure channel exists between one legitimate party and the intruder, and there is another secure channel between the intruder and the second legitimate party. The legitimate parties talk to each other, and the intruder sees all the traffic.

TLS attempts to authenticate the remote party using a *Public Key Infrastructure* (PKI). The idea is to use asymmetric cryptography, where each party has a private key capable of performing cryptographic signatures, and a public key, which can be used to verify a signature done by the associated private key. If a remote party is able to

produce a public key validated signature for a nonce it was given, that proves it has the private key.

We are therefore able to authenticate a machine for which we already know a public key. That leaves a problem to solve: how can we learn the public key for a machine we never connected to?

Enter X.509 certificates, which are also known as SSL certificates. An X.509 certificate is a public key, with data such as the machine name attached, and signed by a private key which is known as a Certificate Authority (CA). When connecting to a server using TLS, the server sends its X.509 certificate. Provided we have the public key of the CA that signed it, we can check the signature and have a hint the public key is the right one for the host we want to connect to, and nobody attempted to perform a MITM attack. Of course if the software has bugs, this check can be incomplete, and this is what happened to Apple [1] and GnuTLS [2] recently.

But this mechanism relies on the following assumption: we have the public key of the CA. How do we have it? For Unix command-line tools such as wget or curl, you need to install it first. This means it cannot work out of the box, but it guarantees a level of security, as you can choose what CA you trust. Note that some distributions provide curl and wget configured with a system-wide CA repository, hence your mileage may vary. Try downloading a well known https:// to discover the behavior of your system.

At the other end of the spectrum, we have web browsers and mail clients, which are bundled with the public keys for hundreds of commercial CA. Since any CA can sign a certificate for any machine name, this means a web browser user trusts hundreds of CA, some of them running in oppressive jurisdictions where a government can compel the CA to sign a certificate for someone else's machine. The CA can also be compromised by hackers [3], with the same result of having someone able to impersonate a machine during TLS authentication.

There is software to help defend against such attacks. For instance, Firefox has a Cert Patrol module that alerts you when the certificate of a web site is not the same as usual. That will let the user spot a MITM attack using a rogue certificate, except of course when connecting for the first time.

There is an even worse attack possible on X.509 certificates, when the cryptography used to build the certificate is too weak and can be cracked. For instance, a 512 bit RSA private key, which was once safe, is now vulnerable to factorization attacks [4], where the private key can be easily derived from the public key by computation. Once the private key is discovered, an attacker can just record TLS traffic and decipher it. We will see later that this can be avoided using Perfect Forward Secrecy (PFS), but even in that situation, an attacker can still decipher the traffic by running a MITM attack.

This leads us to the key length question: how long should it be?

The longer it is, the longer it will take before it can be compromised.

But a key too long means slower cryptographic operations, and perhaps software incompatibility. 1024 bit RSA is the next target, hence you should consider using 2048 bit RSA. 4096 bit RSA is even safer and will work most of the time, but there can be compatibility issues.

Huge key length is not always a guarantee for strong cryptography. Most cryptographic operations need a random source, and if an attacker can predict it, she may be able to decipher data or tamper with it. The private key generation step critically needs a good random generator. If your operating system warns you about low entropy for random source, make sure you fix it before generating private keys. Modern systems feed their random source from entropy gathered from various sources as they run. After a reboot, the entropy pool is low, but the system may be configured to save and restore that information across reboots. In virtualized setups, the virtual machine may be configured to grab entropy on startup from the hypervisor.

Sometimes you will learn about a random generator weakness in your Operating System. This can be the result of an implementation bug, which happened for instance to Debian [5], or a design mistake, for instance in the Dual Elliptic Curve random number generator. In the latter case, the mistake was made on purpose by the NSA, as a document leaked by former NSA contractor Edward Snowden taught us [6]. In both situations, you first need to fix the random number generation process, then regenerate keys, and finally think about what traffic may have been compromised.

While we are on the key factoring problem, it is worth mentioning the Shor algorithm, which allows quick factorization of decent sized keys. Fortunately this algorithm requires a quantum computer to operate, and as of today we are not aware of any implementation of such a system. But if it appears some day, asymmetric cryptography as it is used in TLS will become almost as useless as a Caesar's code [7] is today.

We talked about RSA keys. RSA is indeed the most used public key algorithm. It was invented by Rivest, Shamir and Adleman, hence the name. There are alternatives: DSA, and ECDSA. The latter is not yet widely supported, but it is interesting because it allows much faster operations, which is very valuable on embedded devices.

Key points

- Keep software up to date to avoid running known bugs in certificate validation and weak cryptography.
- Make sure your certificates are signed by a CA known by the client.
- Use certificate tracking software like Cert Patrol on the client.
- Make sure your system random generator is not predictable.
- Use RSA keys for compatibility; if possible, also use ECDSA keys.
- Use key length as long as possible considering performance and compatibility issues.

How-to

- create a 4096 bits RSA private key with appropriate file system permissions (umask 077; openssl genrsa -out private.key 4096)
- create a certificate request (to be sent to a CA for signature) openssl req -new -key private.key -out certificate.csr
- inspect a private key openssl rsa -text -in private.key
- inspect a certificate request openssl req -text -in certificate.csr
- inspect a certificate (signed from the CSR by a CA) openssl X.509 -text -in certificate.crt

Ciphers

TLS uses various algorithms known as ciphers to check data integrity and encrypt it. The specifications make many ciphers available, and implementations do not have to implement them all. The high level of cipher choice means we will have to make sure the right ones are used. This is important because, among ciphers, some are secure for today's standards and some are trivial to crack. There are even null-ciphers, which perform no encryption at all. This is useful for debugging, but of course it should not be used for anything meant to be confidential.

The complete list of available ciphers for OpenSSL-based software can be obtained by running the following command:

```
        openssl ciphers -v
ECDHE-RSA-DES-CBC3-SHA  SSLv3 Kx=ECDH     Au=RSA
   Enc=3DES(168) Mac=SHA1
ECDHE-ECDSA-DES-CBC3-SHA SSLv3 Kx=ECDH     Au=ECDSA
   Enc=3DES(168) Mac=SHA1
SRP-DSS-3DES-EDE-CBC-SHA SSLv3 Kx=SRP      Au=DSS
   Enc=3DES(168) Mac=SHA1
SRP-RSA-3DES-EDE-CBC-SHA SSLv3 Kx=SRP      Au=RSA
   Enc=3DES(168) Mac=SHA1
EDH-RSA-DES-CBC3-SHA    SSLv3 Kx=DH       Au=RSA
   Enc=3DES(168) Mac=SHA1
EDH-DSS-DES-CBC3-SHA    SSLv3 Kx=DH       Au=DSS
   Enc=3DES(168) Mac=SHA1
ECDH-RSA-DES-CBC3-SHA   SSLv3 Kx=ECDH/RSA Au=ECDH
   Enc=3DES(168) Mac=SHA1
ECDH-ECDSA-DES-CBC3-SHA SSLv3 Kx=ECDH/ECDSA Au=ECDH
   Enc=3DES(168) Mac=SHA1
DES-CBC3-SHA            SSLv3 Kx=RSA      Au=RSA
   Enc=3DES(168) Mac=SHA1
PSK-3DES-EDE-CBC-SHA    SSLv3 Kx=PSK      Au=PSK
   Enc=3DES(168) Mac=SHA1
(...)
```

For the curious, each cipher has an IANA-registered number [8] used in the TLS protocol. On recent versions of OpenSSL, that number can be displayed using openssl ciphers -V. Cipher names, in the first column, contains hyphen-separated components. We find here the encryption algorithm, with optional key length. Here are a few examples:

- DES – The ancient Data Encryption Standard, with 56 bit keys
- 3DES – Triple DES, which is equivalent to 168 bit keys
- RC2 – Ancient and insecure Rivest Cipher v2, with 40 bit keys
- AES128 – Modern Advanced Encryption Standard, with 128 bit keys
- AES256 – Modern Advanced Encryption Standard, with 256 bit keys

The key here is different from the private key used in an X.509 certificate. The latter uses asymmetric cryptography, which uses a lot of CPU power, while the former uses symmetric cryptography, which requires much less processing. As the name suggests, symmetric cryptography uses a secret shared by both parties as the key, which may be renegotiated at regular intervals in time. Because cipher keys are of a different nature, their lengths are much shorter than X.509 certificate key lengths, but that does not mean they are insecure.

Then we find the hash algorithm. Here are a few examples:

- MD5 – Ancient and now insecure Message Digest v5
- SHA – Soon to be insecure Secure Hash Algorithm v1
- SHA384 – Modern SHA v2 with 384 bit long hash

The cipher name can also specify the X.509 certificate private key flavor, if the cipher belongs to a specification recent enough to support something else than RSA:

- RSA – Well known Rivest, Shamir and Adleman
- DSS – Digital Signing Signature, used with DSA keys
- ECDSA – Elliptic Curve DSA

There are different families of ciphers and, within a family, a given cipher may have different modes of operation. This may or may not be reflected in the cipher name:

- Stream ciphers, whose last usable TLS cipher is RC4
- Block ciphers, which can operate in various modes: ECB, CBC, CTR, GCM... The relevant modes for TLS are:
  - CBC Older Cipher block chain, which is quite common with TLS
  - GCM More secure Galois/Counter Mode, introduced with TLSv1.2.

Within the block ciphers, CBC mode provides encryption without integrity.

This means a CBC mode cipher always needs a helper Hash Message Authentication Code (HMAC) function to enable integrity. GCM mode does not have this requirement since it provides both encryption and integrity.

And finally, the cipher may negotiate its private keys using a Diffie-Hellman (DH) exchange:

- EDH – Diffie-Hellman (DH) Exchange
- DHE – Ephemeral EDH, which enables PFS (see below)
- ECDH – Newer and faster Elliptic Curve Diffie-Hellman
- ECDHE – Ephemeral ECDH, which enables PFS (see below)

DH exchange support is an optional but very valuable feature. It ensures that the symmetric cipher keys cannot be easily computed from stored TLS exchanges if the X.509 certificate private key is compromised in the future. That feature is called Perfect Forward Secrecy (PFS) or just Forward Secrecy (FS), since nothing is perfect.

It is of no help for the ongoing TLS exchanges done after the X.509 certificate private key is compromised, but it protects stored communications, which is interesting since we know the NSA stores everything it can.

Some clients only implement ECDHE, hence it is desirable to support it; although, it may require quite recent versions of software, as we will see later for Sendmail. For Apache, the latest 2.2.x will support it.

And while we deal with Elliptic Curve cryptography, it is important to note that the algorithm may use different constants over which the administrator has no control. Also, some values may be less safe than others [9], although we do not know practical attacks on that front yet.

Client and server negotiate a common cipher, which should obviously be known by both implementations. On both sides, the software may allow an ordered cipher suite to be configured. For instance, Apache does this through the SSLCipherSuite directive. The client setting usually prevails, unless the server requests otherwise. This is achieved on Apache using the SSLHonorCipherOrder directive.

In OpenSSL-based software the cipher suite syntax is a column-separated list of cipher names, with some additional syntax explained in the openssl_ciphers(1) man page. Here are a few examples:

- ECDH selects all ECDHE-enabled ciphers
- HIGH selects all high security ciphers (128 bit key length and beyond)
- TLSv1 selects all TLSv1 ciphers
- MD5 selects all ciphers using MD5

The openssl ciphers command can be used to see all the ciphers enabled by a particular cipher suite. Here is an example openssl ciphers `ECDH:!RC4:!MD5:!NULL`.

A good cipher suite specification favors the high security ciphers, forbids the insecure, and allows enough ciphers so that any client can connect. Here is a cipher suite specification that favors PFS and ciphers with 128 bit keys and beyond, while remaining compatible with all modern web browsers:

```
  ECDH@STRENGTH:DH@STRENGTH:
HIGH:!RC4:!MD5:!DES:!aNULL:!eNULL
```

Key points

- Make sure server cipher setting prevails over the client's setting.
- Configure cipher suite server-side to disable insecure ciphers.
- For the sake of interoperability, do not restrict available ciphers too much.
- Favor ciphers that can do PFS, and ciphers with longer keys.

How-to

- Compare two cipher lists

```
openssl ciphers ALL:-NULL |tr ':' '\n' > /tmp/1
openssl ciphers ALL:!aNULL:!eNULL |tr ':' '\n' > /tmp/2
diff /tmp/1 /tmp/2|less
```

- Dump all ciphers supported by a service (the sslscan tool is an alternative)

```
for c in `openssl ciphers ALL|tr ':' '\n'`; do
  echo "" |openssl s_client -cipher $c -connect www.
  example.net:443 2>/dev/null
done | awk '/Cipher.*:/{print $3}'
```

Protocols

There are now five versions of the protocol: SSLv2, SSLv3, TLSv1, TLSv1.1 and the latest, TLSv1.2. SSLv2 is very well known to be insecure now and should not be used nowadays. Of course, using the latest version of the protocol is desirable, but it can only be used if both server and client support it. Most of the time it does not happen, which means that a server must support versions down to SSLv3 in order to avoid blocking older clients.

Newer versions of the protocol introduce support for stronger ciphers and fix various vulnerabilities that existed in SSLv3 and TLSv1.

In an ideal world, these two earlier versions of the protocol should be phased out, but unfortunately support for TLSv1.1 and TLSv1.2 is still far from being universal in web browsers; therefore, we need to support older SSLv3 and TLSv1, and work around their security flaws.

As an administrator, there is therefore very little to do with protocol versions. Disable SSLv2, and make sure your software is recent enough to use TLSv1.2. If you happen to develop software that uses TLS, there are a few things to know, though.

When using the OpenSSL library in the C language and initializing SSL contexts, an `SSL_METHOD` must be provided, and this `SSL_METHOD` is obtained by a set of functions that help select the TLS version protocol. Here are the client side flavors (server side flavors look exactly the same):

```
SSLv23_client_method()
SSLv3_client_method()
TLSv1_client_method()
TLSv1_1_client_method()
TLSv1_2_client_method()
```

Using `TLSv1_2_client_method()` looks appealing, but unfortunately this function only enables TLSv1.2, which means that it will prevent from connecting to a server that does not support TLSv1.2. On the other hand, `SSLv23_client_method()` looks undesirable because it may enable insecure SSLv2. But as the function name does not suggest, `SSLv23_client_method()` is able to negotiate the highest protocol version available, up to the latest TLSv1.2 if the server supports it. Using `SSLv23_client_method()` while explicitly disabling SSLv2 (this is done using `SSL_CTX_set_options()` with `SSL_OP_NO_SSLv2`) will therefore bring you the best trade off between security and compatibility. Another common issue is software that does not set up certificate validation. This should be done with the following functions:

```
SSL_set_verify() or SSL_CTX_set_verify()
```

Theses functions take a mode, which should obviously not be `SSL_VERIFY_NONE` for a client that wants to authenticate a server.

```
SSL_set_verify_depth() or SSL_CTX_set_verify_depth()
```

This sets the maximum certificate chain depth when validating the certification chain.

PHP programmers face the same problems. Various functions accept a `ssl://` or `tls://` URL as an argument. Here again, `tls://` looks more modern and more desirable, but in PHP 5.3.x it will translate into `TLSv1_client_method()`, which means only `TLSv1. ssl:// will use SSLv23_client_method()` and enable up to TLSv1.2 if possible.

Using `ssl://` instead of `tls://` seems therefore better, but unfortunately PHP 5.3.x does not configure OpenSSL so that `SSLv23_client_method()` is unable to use SSLv2, hence we have to be sure the server disabled it.

Another PHP 5.3.x issue is that, by default, no certificate validation is done. It is possible to enable it using stream context options, and while there, ciphers can also be specified. Here is an example, with backward compatibility code for older PHP versions (where certificate validation will not be done):

```
if (function_exists('stream_socket_client') {
  $remote = sprintf("ssl://%s:%d", $host, port);
  $opts = array(
    'ssl' => array(
      'verify_peer' => TRUE,
      'verify_depth' => 5,
      'cafile' => '/path/to/ca_file',
      'ciphers' => 'ECDH@STRENGTH:DH@STRENGTH:!RC4:!MD5:!D
   ES:!aNULL:!eNULL',
    ),
  );
  $ctx = stream_context_create($opts);
  $timeout = ini_get("default_socket_timeout");
```

```
  $stream = @stream_socket_client($remote, $errorNumber,
   $errorString,
                                  $timeout, STREAM_CLIENT_
   CONNECT, $ctx);
} else {
  $stream = @fsockopen('ssl://' . $host, $port,
   $errorNumber, $errorString);
}
```

Other languages will often face the same kinds of issues. Generally speaking, it is a good idea to check what happens when connecting to a server with a self signed certificate. If it works without an error, it means some specific code must be added to perform certificate validation. Disabling SSLv2 and using the best protocol version is also often a non-trivial issue that should be inspected.

TLS Configuration hardening in practice: Apache

We will now look at how to harden TLS configuration for a few services: Apache, Sendmail, Dovecot, OpenVPN... Since we talk about examples, we will of course not cover every software available, but once you know what to look at, it is easy to do the same for another program. Our first target will be Apache.

First let us look at what we already covered:

- Make sure your Apache is recent enough to support TLSv1.2 and ECDH: Latest Apache 2.2.x or Apache 2.4.x will do it.
- Generate an RSA private key at least 2048 bits long
- Make sure your certificate is signed by a CA known by browsers
- Use the following configuration for Apache:

```
SSLProtocol all -SSLv2
SSLHonorCipherOrder On
SSLCipherSuite ECDH@STRENGTH:DH@STRENGTH:HIGH:!RC4:!MD5:!
  DES:!aNULL:!eNULL
```

Removing SSLv3 (using SSLProtocol all -SSLv2 -SSLv3) and triple DES (using !3DES in SSLCipherSuite) would be desirable, but it may lock out some older clients. As older devices will get replaced, this will become possible without any drawback. Logging protocols and ciphers selected by your clients is the only way to know if the time has come to phase out SSLv3 and triple DES.

Additionally, if you do not serve clear text content, let the client know it by using the Strict Transport Security HTTP header. Once it has been received by a browser, it will force HTTP over TLS even if the user follows a http:// URL setup on a malicious site. This thwarts possible MITM attacks. Here is how to set it up on Apache:

```
Header set Strict-Transport-Security "max-age=15768000"
```

There is also an older mechanism specific to cookie protection. Cookies are used to maintain sessions on web applications, and they are therefore authentication credentials that should not leak. When setting a cookie, one can set the secure flag, which means the cookie must be sent only over TLS protected connexions. Although redundant for cookie protection, it does not hurt to set up both secure cookies and Strict-Transport-Security.

Cookies can also have the httpOnly flag, which means the JavaScript environment cannot access them. It is a good idea to use it, as it reduces the attack surface on session cookies. Most of the time this requires modifying application code, but there are situations where it can be achieved by the administrator. For instance, applications using PHP sessions can get both secure and httpOnly cookies by setting `session.cookie_secure` and `session.cookie_httponly` to On in php.ini.

Next, test your setup. There is a very useful SSL server test [10] on the web for that, thanks to Qualys SSL Labs. Run the test and it will point out most mistakes in your TLS configuration. If you used the configuration suggested in this paper, you will probably have a bad mark. Why? There can be two kinds of issues.

The first possible problem, is that you read this paper a long time after its publication, and the sample configuration is now insecure. You will have to adjust it with the help of Qualys SSL server test. If your certificate key length or hashing algorithm is now weak, improve it with better settings. If the tests say a TLS version is now insecure, adjust the SSLProtocol to remove it. If you have a problem with ciphers, look at the test results for a browser handshake simulation. Check what cipher is picked by each browser and adjust the SSLCipherSuite.

Once you fixed the certificate, protocol and cipher, you may still get a bad mark despite the TLS configuration being pretty good. This is because the web is a quite complex environment, which makes some web-specific attacks against TLS possible; hence we need to work around them.

Mitigating the CRIME attack

The CRIME [11] attack belongs to the chosen plaintext class of attacks. It allows the attacker to extract authentication cookies from the TLS-protected data stream, which allows session hijacking. The CRIME attack can happen if:

- the browser is tricked into sending data controlled by the attacker
- the attacker can observe the server response
- TLS compression is enabled

The first two conditions can happen if the attacker controls content from the served pages, which can happen when the server has Cross Site Scripting (XSS) vulnerabilities. The other situation where it may happen are cross-site requests, where a malicious web site tricks the browser into sending data to the attacked web site.

The only CRIME workaround is to work on the third condition, and to disable TLS compression. This is the default in recent Apache releases, as in recent browsers. On earlier Apache releases, the following configuration directive lets you disable it:

```
SSLCompression  off
```

Disabling TLS compression does not affect performance a lot, since on a well configured server, HTTP compression is enabled. But as we will see in the next section, it also brings problems.

Mitigating the BREACH attack

The BREACH [12] attack is similar to CRIME. Here the offending feature is not TLS compression but HTTP compression. A quick fix is to disable HTTP compression, but that could have a bad impact on performance.

A nice solution is to disable HTTP compression for requests coming from another web site. That will not help if the attacker uses an XSS vulnerability, but there are many other ways to attack your site in that scenario; hence we will assume you already fixed it.

On the other hand, that prevents an attacker from running a BREACH attack from another web site. This workaround can be implemented this way in Apache, assuming your server name is *www.example.net*:

```
      # BREACH mitigation
      SetEnvIfNoCase Referer .* self_referer=no
      SetEnvIfNoCase Referer ^https://www\.example\.net/
self_referer=yes
      SetEnvIf self_referer ^no$ no-gzip
```

Mitigating the BEAST attack

The BEAST attack works on SSLv3 and TLSv1, against a class of cipher known as block ciphers (most of the time they have CBC, like Cipher Block Chaining, in their names). The obvious fix is to disable block ciphers, but that leaves us with the following ciphers available:

- GCM (Galois/Counter Mode) ciphers
- RC4

GCM ciphers are only available with TLSv1.2, which means it does not help since we look for a mitigation against an attack on SSLv3 and TLSv1.

RC4 could help and, at some time, it was advised to craft an SSLCipherSuite setting that favored GCM ciphers, then RC4, so that newer browsers could pick GCM and older would use RC4. Here is an example:

```
ECDH@STRENGTH:DH@STRENGTH:-SHA:ECDHE-RSA-RC4-
   SHA:HIGH:!MD5:!DES:!aNULL:!eNULL
```

Unfortunately, RC4 itself was discovered to be weaker than previously thought [13], leaving system administrators with the choice between two evils: be vulnerable to a RC4 attack that requires hours of exchanges, or be vulnerable to BEAST attacks.

The former alternative may look more desirable, but web browsers made progress at defending against BEAST, either by supporting TLSv1.1 or TLSv1.2, or by implementing a hack called the 1/n-1 split, that makes TLSv1 CBC ciphers not vulnerable to BEAST. On the other hand, there is no way to defend against attacks on RC4, and they are likely to get more efficient as time goes.

Favoring RC4 hence improves the security for browsers still vulnerable to BEAST, but it decreases it for browsers that implemented BEAST workarounds and would be pushed to pick a weak RC4 cipher instead of for instance AES256.

This is the reason why today it makes sense to disable RC4 and favor BEAST-vulnerable, PFS-capable block ciphers. This is what we did in the suggested SSLCipherSuite in this paper, but there is no broad consensus on this: Some websites still prefer to favor RC4.

This is the case for Google as of May of 2014. It makes some sense because an attack on RC4 still requires a lot of exchanges as of today.

Despite BEAST mitigation being widely available, you may have clients that have not been updated and are still vulnerable.

If you chose a server configuration that promotes strong ciphers but leaves older clients vulnerable to BEAST (i.e.: you did not favor RC4), you may want to detect vulnerable clients and report the problem to the user in order to push for a browser update.

The author of this paper wrote an Apache module for that [14]. Qualys SSL Labs also has a more feature-rich browser fingerprinting Apache module that can detect BEAST [15].

Session renegotiation

In 2009, a vulnerability was discovered in a feature called session renegotiation [16]. As the name suggests, it allows the change of TLS session parameters. For instance, you could have a part of a website that wants the client to authenticate using an X.509 certificate, while the remainder of the site could be accessed without a certificate.

When the client connects to the web server and negotiates the TLS session, the server does not know the requested URL yet. In order to enforce a per-URL policy for requiring client certificates, the server needs to renegotiate the TLS session after the URL has been sent. This is that exact feature that has been under attack. The TLS renegotiation procedure has been updated in the TLS protocol to work around the problem.

The only reasonable fix here is to upgrade the server so that it supports secure TLS session renegotiation. This will not be backward compatible with older browsers that have not been updated. If you need to support them, you will have to give up on the per-URL policy, and use multiple virtual servers using different IP addresses. That way Apache knows what to do before the TLS handshake is done and TLS renegotiation is not needed anymore.

Fixing Heartbleed

Heartbleed is the TLS attack that got the most press coverage, thanks to the communication efforts of the researchers that discovered it: a good name, a dedicated web site, and perhaps a first time for a computer vulnerability, a dedicated logo. Unfortunately, the vulnerability itself is extremely severe, making the efforts to get it picked up by mainstream press legitimate.

As opposed to protocol vulnerabilities like CRIME, BEAST and BREACH, Heartbleed is an implementation vulnerability: just a programming error in OpenSSL that introduces a vulnerability in a TLS extension called TLS Heartbeat. Other TLS implementations such as GnuTLS or Mozilla NSS are not vulnerable, neither are OpenSSL 0.9.x releases and earlier, which did not have the TLS Heartbeat functionality.

The bug is a buffer overflow that allows an attacker to read 64 KB of memory from the server, which can include various data recently used, like session cookies, passwords, database access credentials, and, in the worst case, the server private key (the only case where the private key remains safe while the server is vulnerable is when it is stored in a hardware security module because it never appears in memory).

Buffer overflow mitigation techniques such as *Address Space Layout Randomization* (ASLR) are not effective against Heartbleed because OpenSSL implemented its own memory management on top of libc's one, for performance's sake. That situation is controversial since it is not granted that OpenSSL's memory management layer is faster than libc's on any one system. Even on systems where it is, this is a problem that should be fixed in libc itself, and not worked around in OpenSSL.

Even more unfortunate is the fact that TLS Heartbeat is an optional extension that is almost useless. It is used to keep connexions alive in Datagram TLS (DTLS), which is TLS over UDP. When TLS runs on top of TCP, which is the case for HTTP over TLS, the problem solved by Heartbeat does not exist.

Since Heartbleed is only an implementation vulnerability, fixing it means upgrading OpenSSL to at least version 1.0.1g, or rebuilding it without TLS Heartbeat enabled (that is, using `-DOPENSSL_NO_HEARTBEATS` option to the C compiler). Since the vulnerability may have allowed private keys to be extracted, they must be replaced, TLS certificates must be renewed and the older certificate should be revoked. Then all user passwords should be changed. All that work is painful, but probably necessary.

There has been some controversy on how easily the server private key could be extracted. The vulnerability only allows reading 64 KB of data, and most of the time the private key is out of reach, but it has been shown that it can be reliably obtained just after a server restart [17]. Reliable or not, a private key-leak vulnerability is still a major concern, as one leak is enough to compromise the private key forever. And moreover, the vulnerability has existed for two years before being disclosed and fixed, hence it cannot be excluded that someone knew it and used it during that time frame.

Ineffective certificate revocation vulnerabilities

There is a dark corner of Heartbleed recovery that has not been covered a lot: many web browsers never check for certificate revocation, and, even if they do, there are situations where they fail to do it and still allow the connexion to proceed (WiFi hotspot is a situation where this happens). This means that if an attacker stole a private key, he is still able to use it with the revoked certificate to perform Man in the Middle attacks. This means an effort will have to be made to improve certificate revocation enforcement.

There are two widely supported methods to check for certificate revocation. The first one is Certificate Revocation List (CRL), a CA-signed list of revoked certificate serial numbers, which is published at a well known URL carved into the CA certificate. This method obviously does not scale, as the file length only increases as certificates are revoked. And it also requires a network connexion.

The other major approach is *Online Certificate Status Protocol* (OCSP), a protocol that lets the browser ask for the revocation status of a particular certificate. It scales much better than CRL, but it still requires a network connexion to be used.

Google introduced an offline mechanism in Chrome, which is called CRL sets. The idea is that Chrome gets CRL with its automatic updates, removing the need for a network connexion when a certificate revocation status is checked. Unfortunately, this mechanism suffers a scaling problem even worse than CRL, as CRL sets should list revoked certificate serial numbers for all the CA known to the browser. This would be huge, and as a result, Google only provides CRL sets for a subset of the installed CA [18], leaving the user vulnerable without any clue about which web site is protected by CRL sets and which one is not.

The problem could be addressed correctly by OCSP stapling, which is a TLS extension that enables the TLS server to acquire a certificate validity proof from the CA so that it can be presented to the client during the TLS handshake. In other words, the TLS server acts as an OCSP proxy for the client. This solves both the scalability problem (except perhaps for the network bandwidth consumed by OCSP services) and the network access requirement: a not-yet-authenticated host on a WiFi hotspot is able to assess the authentication server's certificate revocation status. But unfortunately, OCSP stapling requires client and server support, and is not yet widely supported.

Other attacks

There are other attacks against TLS which the Qualys SSL server test will not warn you about. The padding oracle attack was discovered in 2002, and it was worked around for a long time by all implementations, but a new usable variant called Lucky 13 was published in 2013.

Like BEAST, Lucky 13 targets CBC ciphers, which means it can be mitigated by using RC4 (but this trades one vulnerability for another), or by using TLSv1.2, which brings GCM ciphers. There is also a Lucky 13 implementation workaround for CBC ciphers, obtained by introducing time variations in the algorithms. But both GCM and CBC with time variations are not implemented by all clients, and there is no way for the server to know if Lucky 13 mitigation for CBC is done on the client or not.

Fortunately, Lucky 13 needs the attacker to tamper with TLS traffic, which means it must be in a position to act as a Man in the Middle (MITM). And since it is a timing dependent attack, it requires a network connexion with low jitter (jitter is the variation of latency) and a lot of requests to succeed and recover a session cookie. Hence, the risk is low for now. It can be lowered by limiting session cookie lifetime or, even better, by binding it to the number of requests done.

Client vulnerabilities

As noted earlier, some TLS vulnerabilities require client fixes. This is the case for TLS session renegotiation, and BEAST, through the 1/n-1 split workaround in CBC ciphers. Lucky 13 also is spared by implementing timing variation for CBC ciphers. If we assume the client has not been updated, we can have server-side mitigation by avoiding TLS session renegotiation, or by using RC4 (but as already said it is a controversial choice, since RC4 is vulnerable to other attacks) or GCM ciphers available in TLS 1.2 (which is still not supported by many clients). However, there are other threats against clients for which no server-side mitigation is available.

The Apple TLS stack suffered a security failure because of a duplicate "goto fail" statement in the source code. It allowed easy hijacking of TLS session for TLS up to TLSv1.1 and ciphers using ECDHE or DHE (these are the ciphers required to support Perfect Forward Secrecy).

This caused a widespread vulnerability on iOS, and vulnerabilities in Safari and Apple Mail on MacOS X. The only decent workaround is to upgrade the OS. On MacOS X, software that brings their own TLS implementation, such as Firefox and Thunderbird, could also be used.

As for the BEAST vulnerability, identifying the browsers vulnerable to the "Apple goto fail" vulnerability is a good idea, since it allows users to be notified. This can be done by running a dedicated Apache linked with a patched OpenSSL, as described by Qualys SSL labs' Ivan Ristic [19].

If you have a corporate intranet start page, it is a good place to run Multiple browsers vulnerabilities tests. This can easily be achieved by including `<img>` tags for 1x1 images on a test server. If a vulnerability is identified, a site-wide cookie can be set and the corporate intranet start page can use it to warn the user.

While there, you can also test if the browser accepts self-signed certificates, which would allow anyone to perform easy Man in the Middle attacks. Some mobile browsers fail to perform this basic check, which means their TLS implementation does not bring much more security than clear text communications.

Key Points

- Use up to date server software
- Test your TLS-enabled web server with Qualys SSL server test
- Make sure TLS compression is disabled

- Disable HTTP compression for requests incoming from other sites
- Until the day TLSv1.2 support becomes widespread, choose between dangers of RC4 and CBC ciphers.
- Assess browser vulnerabilities and notify users so that they know they need an upgrade
- Protect session cookies with the secure and `httpOnly` flags

Hardening other protocols

Once your web servers are secure, it is time to look at other protocols. Here are a few obvious examples: POP, IMAP, SMTP, LDAP, OpenVPN.

The good news is that many attacks possible on HTTP over TLS cannot be transposed to other protocols. This is because there is no way to trick the client into sending chosen data.

POP requests cannot bounce from an external server to yours, and no JavaScript can inject data into an SMTP session.

This is true for all above mentioned protocols. It means attacks on CBC ciphers such as BREACH, CRIME or BEAST cannot happen here. The only thing we will have to worry about beyond certificates, protocols and ciphers are Lucky 13 and RC4 vulnerabilities.

Dealing with RC4 vulnerability is simple, we can just disable it. This is an easy choice here since we do not choose between RC4 attacks and BEAST.

On the other hand, protecting against Lucky 13 is not easy. We cannot enforce the use of TLS 1.2 GCM ciphers, as many clients do not support them, and we do not want to use RC4. We may assume timing variant workarounds to be implemented in the client, we have no way to assess it is the case.

Unfortunately, this client workaround and the difficulty to run a Lucky 13 attack are our only solutions against Lucky 13 until the day TLS 1.2 is universally supported.

Dovecot

Here is a sample dovecot TLS setting:

```
ssl = yes
ssl_ca = </etc/openssl/certs/ca-chain.crt
ssl_key = </etc/openssl/private/server.key
# Server certificate with CA chain included
ssl_cert = </etc/openssl/certs/server-bundle.crt
# No need to disable SSLv2, it is done by default
ssl_prefer_server_ciphers = yes
ssl_cipher_list = ECDH@STRENGTH:DH@STRENGTH:HIGH:!RC4:!M
 D5:!DES:!aNULL:!eNULL
disable_plaintext_auth = yes
```

Note that ECDHE is available starting with Dovecot 2.2.6. As for the web server, supporting it is important because it will increase Perfect Forward Secrecy capable clients a lot. SSLv2 is supposed to be disabled by default, but it does not hurt to check that:

```
openssl s_client -ssl2 -connect server:993
```

Sendmail

Many TLS features in Sendmail need to be enabled at compile time. This is achieved by filling the site.config.m4 before the build:

```
# enable STARTTLS
APPENDDEF(`conf_sendmail_ENVDEF', `-DSTARTTLS')
APPENDDEF(`conf_sendmail_LIBS', `-lssl -lcrypto')
# enable _FFR_TLS_1, for CipherList directive
APPENDDEF(`conf_sendmail_ENVDEF', `-D_FFR_TLS_1')
# enable _FFR_TLS_EC, for ECDH support, requires sendmail
  8.14.8 or above
APPENDDEF(`conf_sendmail_ENVDEF', `-D_FFR_TLS_EC')
```

If your sendmail was obtained as a binary package, the following command lets you know what is built inside (look for STARTTLS, _FFR_TLS_1 and _FFR_TLS_EC):

```
sendmail -d0.13 < /dev/null
```

`_FFR_TLS_EC` was contributed by the author of this paper. It is really new and is not likely to be built in a binary package in the months following the publication of this article. Also note that FFR stands for “For Future Release”, which suggests that the feature may become default in the future. Once you have the correct options, either because you already had them, or because you made a custom sendmail build, you can use the following options in sendmail.cf:

```
O CACertPath=/etc/openssl/certs/
O CACertFile=/etc/openssl/certs/ca-chain.crt
O ServerCertFile=/etc/openssl/certs/server.crt
O ServerKeyFile=/etc/openssl/private/server.key
O DHParameters=/etc/openssl/certs/dh.pem
O CipherList=ECDH@STRENGTH:DH@STRENGTH:HIGH:!RC4:!MD5:!DE
  S:!aNULL:!eNULL
O ServerSSLOptions=+SSL_OP_NO_SSLv2 +SSL_OP_CIPHER_
  SERVER_PREFERENCE
O ClientSSLOptions=+SSL_OP_NO_SSLv2
```

Recent sendmail will not need the DHParameters option, but it is worth a few words. ECDHE and DHE ciphers need DH parameters to operate. Some software is able

to auto-generate them internally, while others will require the administrator to generate a DH parameter file and specify it in the configuration. The file can be obtained by running the following command:

```
openssl dhparam 2048 > /etc/openssl/certs/dh.pem
```

OpenLDAP

TLS support in OpenLDAP has been good for a long time, hence you are not likely to have missing features.

There are two cases here: the TLS service available to clients and the TLS connexions between master server and replicas. The former needs to be permissive enough so that any client can connect, but the latter is a good candidate for hardening since it only needs to support connexions from replicas using recent OpenLDAP releases.

Here is a hardened configuration for an OpenLDAP master:

```
TLSCertificateFile      /etc/openssl/certs/server.crt
TLSCertificateKeyFile   /etc/openssl/private/server.key
TLSCACertificateFile    /etc/openssl/certs/ca-chain.crt
TLSDHParamFile          /etc/openssl/certs/dh2048.pem
TLSCipherSuite          ECDH:DH:!RC4:!SHA:!MD5:!DES:!aN
 ULL:!eNULL
TLSVerifyClient         allow
TLSCACertificatePath    /etc/openssl/certs
TLSCRLCheck             all
```

`TLSCACertificateFile`, `TLSVerifyClient`, `TLSCACertificatePath`, and `TLSCRLCheck` are useful if you allow clients to authenticate using certificates, which is a good idea for LDAP replicas connecting to an LDAP master (more on that later).

Note that !SHA:!MD5 in the CipherSuite will disable all ciphers from TLSv1.1 and earlier: SHA in a cipher specification stands for SHA1, and only TLSv1.2 brings ciphers using hash algorithm using SHA2.

The specification may be odd, but it is just because it is an evolution from the previous cipher suite we used before. The same result could be achieved with something perhaps more easy to understand such as:

```
ECDH:DH:!TLSv1:!SSLv3:!aNULL:!eNULL
```

On the replica side, the cipher suite needs to allow more clients to connect. You can reuse the cipher suite we used for the web, or if your LDAP clients are good enough, you can force PFS usage:

```
TLSCertificateFile      /etc/openssl/certs/server.crt
TLSCertificateKeyFile   /etc/openssl/private/server.key
TLSCACertificateFile    /etc/openssl/certs/ca-chain.crt
TLSDHParamFile          /etc/openssl/certs/dh2048.pem
TLSCipherSuite          ECDH@STRENGTH:DH@STRENGTH:!RC4:!
 MD5:!DES:!aNULL:!eNULL
TLSVerifyClient         allow
TLSCACertificatePath    /etc/openssl/certs
TLSCRLCheck             all
```

OpenLDAP replicas can also have a TLS setup for their client-side, in chain overlay and syncrepl configuration. Here is a configuration sample:

```
syncrepl rid=17
    provider=ldap://ldap-master.example.net
    type=refreshAndPersist
    searchbase="dc=example,dc=net"
    starttls=critical
    schemachecking=off
    sizelimit=unlimited
    retry="3 1 10 2 60 +"
    bindmethod=sasl
    saslmech=EXTERNAL
    tls_cert=/etc/openssl/certs/server.crt
    tls_key=/etc/openssl/private/server.key
    tls_cacert=/etc/openssl/certs/ca-chain.crt
    tls_reqcert=demand
    tls_cipher_suite=ECDH:DH:!RC4:!SHA:!MD5:!DES:!aNULL:!
   eNULL
    tls_cacertdir=/etc/openssl/certs
    tls_crlcheck=all
```

Both the master and replica configuration here include directives to enforce CRL checking. OpenSSL is a bit odd here and deserves some explanations. If CRL checking is enabled using `TLSCRLCheck` (`tls_crlcheck` in syncrepl configuration), then OpenSSL will look for a CRL file in the directory specified by TLSCACertificatePath (`tls_cacertdir` in syncrepl configuration).

The name of the CRL file is of the form `${hash}.r0`, where `${hash}` is obtained by the openssl crl command:

```
openssl crl -hash -noout -in ca.crl
```

This means that once you have obtained the ca.crl file, you must install it in the directory specified by `TLSCACertificatePath` (`tls_cacertdir`), and run the following command:

```
ln -s ca.crl `openssl crl -hash -noout -in ca.crl`.r0
```

If you manage your own internal CA, which is a good idea for LDAP certificates, the CRL file can be obtained

from the CA key and certificate by using this command:

```
openssl ca -keyfile ca.key -cert ca.crt -gencrl -out ca.crl
```

Webmail software

We dealt with web and mail servers, but not with the middleware that can sit between them, which is known as webmail. Many types of software are available, but a close inspection of a few of them (SquirrelMail, RoundCube, ImapProxy) has shown a few deficiencies: no certificate validation was done, no effort was done to push TLSv1.2 usage if available, and cipher suites were not configurable.

While the two latter issues can be acceptable, the lack of certificate validation can be a real problem, as it allows easy Man in the Middle attacks. Often the risk is neglected because mail and web servers sit close to each other in the same data center, but that may not always be the case.

Patches have been submitted and accepted in the three above mentioned types of software to improve the situation but as of today formal releases including the patches have not been done. Here are the minimum version requirements and configuration snippets to get support up to TLSv1.2, ECDHE ciphers, CA validation, and configurable cipher suites:

```
- ImapProxy>1.2.7 (Use SVN version)
        tls_ca_file  /etc/openssl/certs/tcs-chain.crt
        tls_ciphers ECDH:DH:!RC4:!SHA:!MD5:!DES:!aNULL:!eNULL
        tls_verify_server true
        force_tls true

- Squirrelmail>1.4.22 (Use 1.4.23-svn or 1.5.2-svn
   versions)
        $use_smtp_tls = true;
        $smtpOptions['ssl']['verify_peer'] = true;
        $smtpOptions['ssl']['verify_depth'] = 3;
        $smtpOptions['ssl']['cafile'] = '/etc/openssl/
   certs/ca-chain.crt';
        $smtpOptions['ssl']['ciphers'] =
            'ECDH:DH:!RC4:!SHA:!MD5:!DES:!aNULL:!eNULL';
        $smtp_stream_options = $smtpOptions;

- Roundcube>=1.0rc
        $rcmail_config['smtp_conn_options'] = array(
          'ssl' => array(
            'verify_peer' => TRUE,
            'verify_depth' => 3,
            'cafile' => '/etc/openssl/certs/ca-chain.crt',
            'ciphers' => 'ECDH:DH:!RC4:!SHA:!MD5:!DES:!aNU
   LL:!eNULL';
          ),
        );
```

There are a few points to note:

- SquirrelMail and RoundCube certificate validation and cipher specification use PHP context options, which are only available in PHP 5.
- We assume ImapProxy usage here: IMAP configuration for SquirrelMail and Roundcube is therefore not using TLS.
- We assume the mail server TLS setup is under our control and that we can enforce a restrictive cipher suite, with only PFS-enabled TLSv1.2.

OpenVPN

The case of OpenVPN could be straightforward: it enjoyed very good TLS support for a long time, and we could assume clients are up to date.

This would let us enforce the use of TLSv1.2 GCM ciphers. Here are the TLS related options in server configuration:

```
ca /etc/openssl/certs/ca-chain.crt
cert /etc/openssl/certs/server.crt
key /etc/openssl/private/server.key
dh /etc/openssl/certs/dh2048.pem
tls-cipher ECDH:DH:!RC4:!SHA:!MD5:!DES:!aNULL:!eNULL
client-cert-not-required
```

But unfortunately there are many outdated clients that will not support such a setup. TLSv1.2 is not available in OpenVPN up to 2.3.2, and MacOS X's package GUI for OpenVPN, which is known as Tunnelblick, is bundled with various versions of OpenVPN (as of May 2014, the latest Tunnelblick ships with OpenVPN versions 2.2.1, 2.3.2, and 2.3.4). The preference panel lets the user choose the OpenVPN version, and, if it is not set to 2.3.4 or Tunnelblick is too old to include 2.3.4, the client will not work with a TLSv1.2 only configuration. The setup below is more compatible by allowing CBC ciphers from protocol versions prior to TLSv1.2:

```
ca /etc/openssl/certs/ca-chain.crt
cert /etc/openssl/certs/server.crt
key /etc/openssl/private/server.key
dh /etc/openssl/certs/dh2048.pem
tls-cipher ECDH:DH:!RC4:!MD5:!DES:!aNULL:!eNULL
client-cert-not-required
```

More protocols

There may be other protocols to look at. To name a few, RADIUS configuration for EAP-TTLS in 802.1x authentications, SIP, or any TCP service wrapped into stunnel.

This latter utility is worth mentioning, as it allows any TCP stream to be protected by TLS. If you encounter a type of software with broken TLS that cannot be fixed,

disabling its TLS support and replacing it with stunnel is a good idea: the broken software listens for plain text traffic on 127.0.0.1, and stunnel offers the TLS-protected service over the network and relays it to the broken software. The stunnel configuration will look like this:

```
chroot = /var/chroot/stunnel/
setuid = stunnel
setgid = stunnel
pid = /stunnel.pid
debug = 0
CAfile = /etc/openssl/certs/ca-chain.crt
sslVersion = TLSv1
ciphers = ECDH@STRENGTH:DH@STRENGTH:HIGH:!RC4:!MD5:!DES:
 !aNULL:!eNULL

[https]
# contains CA chain and certificate
cert = /etc/openssl/private/cert-bundle.crt
key =  /etc/openssl/private/server.key
accept = 443
# protected service is told to listen on port
 127.0.0.1:8080
connect = localhost:8080
```

Note that such a setup is also useful for debugging TLS-protected protocols. Network sniffers such as tcpdump are of little interest when facing TLS protected streams, but, if you install a stunnel relay, it is possible to capture the clear text protocol by listening on the local interface:

```
tcpdump -ni lo0 -s0 -X 'port 8080'
```

Conclusions

We had a look at TLS and how to harden it against the usual attacks against it for various types of open source software. The method is always the same: make sure software is up to date, disable SSLv2, choose a strong cipher suite that will still allow older clients to connect, and test the result.

Server testing should make sure that insecure protocol versions and ciphers are disabled. Proper certificate validation should also be checked, as this step is sometimes missing in some software. It should be done for clients and for servers that perform client authentication using X.509 certificates. Tests may spot broken software. If it cannot be fixed, using an up-to-date stunnel as a relay is an option to obtain a properly hardened TLS service.

The final words could be an emphasis on the TLS flaws that cannot be worked around. Only TLSv1.2 with recent ciphers is safe from any known attack, and since some clients do not support it, the administrator often needs to choose between two kinds of trouble: on the one hand, using CBC ciphers that are subject to various attacks that may be fixed in the client (BEAST, Lucky 13), and/or difficult to perform (Lucky 13); on the other hand, adopt the weak RC4, which is vulnerable to attacks difficult to perform, but with no fix available, either on the client or the server. No consensus exists here, and as various actors made different choices, the author of this paper chose CBC. Unfixed TLS flaws also exist in older clients and servers, and will certainly happen again in the future. Even TLS-unrelated flaws can have an impact on TLS, for instance when a server private key is compromised because a cracker got root access on a server or managed to derive it from the public key. In such a scenario, all TLS traffic that has been captured and stored in the past can be deciphered, except if a PFS cipher was used.

Acknowledgments

Thanks to Jean-Yves Migeon, Marc Dreyfus and Fredrik Pettai for reviewing this paper.

References

[1] *https://threatpost.com/*
[2] *http://arstechnica.com/security/2014/03/critical-crypto-bug-leaves-linux-hundreds-of-apps-open-to-eavesdropping/*
[3] *http://www.computerworld.com/s/article/9219663/Hackers_may_have_stolen_over_200_SSL_certificates*
[4] *http://en.wikipedia.org/wiki/RSA_numbers#RSA-210*
[5] *https://lists.debian.org/debian-security-announce/2008/msg00152.html*
[6] *http://www.reuters.com/article/2013/12/20/us-usa-security-rsa-idUSBRE9BJ1C220131220*
[7] *http://en.wikipedia.org/wiki/Caesar_code*
[8] *http://www.iana.net/assignments/tls-parameters/tls-parameters.xhtml#tls-parameters-4*
[9] *http://safecurves.cr.yp.to/*
[10] *https://www.ssllabs.com/ssltest/*
[11] *http://en.wikipedia.org/wiki/CRIME_%28security_exploit%29*
[12] *http://en.wikipedia.org/wiki/BREACH_%28security_exploit%29*
[13] *https://community.qualys.com/blogs/securitylabs/2013/03/19/rc4-in-tls-is-broken-now-what*
[14] *https://ftp.espci.fr/pub/mod_logbeast/*
[15] *https://github.com/ssllabs/sslhaf/*
[16] *http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2009-3555*
[17] *http://arstechnica.com/security/2014/04/heartbleed-vulnerability-may-have-been-exploited-months-before-patch/*
[18] *https://www.imperialviolet.org/2014/04/29/revocationagain.html*
[19] *http://blog.ivanristic.com/2014/03/building-your-test-for-apple-tls-auth-bug.html*

EMMANUEL DREYFUS

Emmanuel Dreyfus works as a system administrator in Paris, France. He is a contributor to several open source software, including NetBSD since 2001 and milter-greylist since 2006. His complete list of publication about NetBSD development and Unix system administration can be found at http://hcpnet.free.fr/pubz/.

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

Setting up Your own Package Cluster

in MidnightBSD

During the development of MidnightBSD, a method was needed to test building software packages for mports developers. Packages needed to be built in a clean environment so that they would work across systems.

What you will learn…

- The history of the Magus package software and why it was written
- Installing Magus
- Using Magus

What you should know…

- How to install MidnightBSD or download a virtual machine image
- Usage of mport and the ports tree
- Setting up a PostgreSQL database

In 2008, The MidnightBSD project had a cluster of 21 Dell desktops available in a computer lab at Eastern Michigan University. The individual systems were slow, but there was a lot of computing power available.

Chris Reinhardt wrote the first version of Magus, the MidnightBSD package cluster, over a few months using MySQL and Perl.

The goal was to build a package on each individual computer by determining what other packages were necessary and scheduling intelligently. Each node would lock the port by writing a record to the database along with build logs and status.

Improvements to the system were developed including a web application to monitor the build, scripts to manage the build, and the ability to build in parallel on one system.

In May, Magus was ported from MySQL to PostgreSQL in order to take advantage of some performance advantages with the workload and the author's preference with licensing.

Installing Magus

Magus requires an installation of PostgreSQL 9.2, Perl 5.14, and a checkout of mports from subversion.

To install the Magus client dependencies, install mports/ports-mgmt/magus-utils. This includes all Perl modules needed to run Magus and the database libraries. If you wish to use the Magus web application, you should also install a web server capable of running CGI scripts such as Apache or lighttpd.

Magus source code and configuration examples are located in mports/Tools/magus.

Configuration

Before you begin configuring the system, determine how many magus nodes you will start with in your cluster, the location where packages will be written (such as an NFS share or ssh server), and which node will be the master. A master node will store the packages and mports tarball file generated by the indexing process discussed later.

Magus requires a chroot tarball containing system files. This is used to create a base directory and then during the run, a fresh build directory is created on each node. A script is provided to generate the tarball, `make_chroot_tarball.pl`. You will need to generate a file for each OS version and architecture that you wish to support. The chroot file must be copied to each node inside the build directory, commonly `/usr/magus`.

To configure PostgreSQL, create a new database called magus and appropriate user credentials. Load the schema file `schema.sql` into PostgreSQL. Magus requires knowledge of each node that will be used to build software. Open a connection to the database with a SQL client and add rows for each node in the machines table with OS version, architecture and node name. This will allow

Magus to track which machine is building which port and allow nodes to get work once started.

Listing 1. *Magus Initialization*

```
# SQL Setup
psql magus < schema.sql
# Add a new row for each node you wish to use, arch
   can be i386 or amd64
psql
INSERT INTO machines (arch,name, maintainer,
   osversion,run) VALUES('amd64','master','me@example.
   com', 0.5,0);
\q
```

The configuration file provides the database credentials, path to package files shared by all nodes and the node name. It is used by the slave build script and all administration scripts provided with Magus. A sample configuration file, `config.yaml.example`, is provided and should be placed in the desired build location, `/usr/magus/config.yaml`. On each node, configure the file with the node name, database credentials and server address, path to the chroot tarball and the PkgfilesRoot path to write packages.

Starting Magus

To start the Magus cluster, a new run must be created. A run is a batch of packages and will have a unique ID. You can use the run ID with several of the administrative scripts. On the master node, run the script `mports/Tools/magus/master/update_cluster.pl` to generate a new run. You will specify the OS version and architecture and a new checkout of the ports tree will be created. The generated file will be used by all the Magus nodes during the build so that they have the same version of the mports tree. You can customize the contents of the tar file as well as the chroot tar file to change how ports are built such as turning on SSL or X11 support, etc. Remember that if you add any new ports to the file, you will need to create the relevant entries in the Magus database.

Table 1. *Magus administration scripts*

Script	Description
delete_result.pl	Delete an individual port result so that it may be built again.
nuke_internals.pl	Delete internal errors to rebuild again in case of disk space or other temporary errors.
top_blockers.pl	Investigate failures that prevented other ports from building
update_cluster.pl	Create a new Magus run to build packages

On the Web

- *http://www.midnightbsd.org/* – MidnightBSD Project website,
- *http://www.midnightbsd.org/magus/* – Magus,
- *http://svn.midnightbsd.org/svn/mports/trunk/Tools/magus/* – Magus source code.

Glossary

- magus
- mports

To start the build process, run the script located at `mports/Tools/magus/slave/magus.pl`. This step can be repeated on each node you wish to build with to complete the run.

Use the Magus web application to view current builds and progress. You will see ready ports drop to zero when the build is complete.

Generate a custom index for mport

To build a custom mport index, run the bless program with the run ID and path to the files on the master node. Bless is used to mark a Magus run as complete and ready for mass consumption. The index can be used on systems to provide a list of packages available and is required for proper operation of the mport package management tool. You can also provide a list of servers in the index file to fetch packages from over HTTP.

Future Directions

Magus has recently been ported to PostgreSQL from MySQL. We would like to create a RESTful web service for nodes to interact with the build cluster. This would allow nodes to be distributed without the need to access a database server directly.

The build software needs to be modified to disconnect fetching distribution files from the build process to avoid fetching the same source code multiple times and to speed up cluster builds.

Summary

Magus is the package building software for MidnightBSD mports. It allows system administrators managing multiple systems to generate custom packages for their own systems. It also allows developers to test changes to mports without committing code to the subversion repository.

LUCAS HOLT

Lucas Holt is the founder of the MidnightBSD project and a Senior Application Programmer/Analyst for the University of Michigan in Ann Arbor, MI, USA.

Getting to Grips with the Gimp – Part 5

In the fifth part in our series on the Gimp we will learn about layer masks and how to modify faces and hair colour.

What you will learn…

- How to manipulate images like a design pro

What you should know…

- General PC administration skills

Professional graphic designers are often called to retouch images. In this tutorial, we will take this one step further and transplant the face of one model onto another. The trickiest part of this exercise is matching the skin tone and making the appearance realistic. While no humans were harmed in the creation of this tutorial, my apologies to the models concerned – you look far better in your original images!

Part 1 – Perform a face transplant

Step 1
Download the images from Table 1.

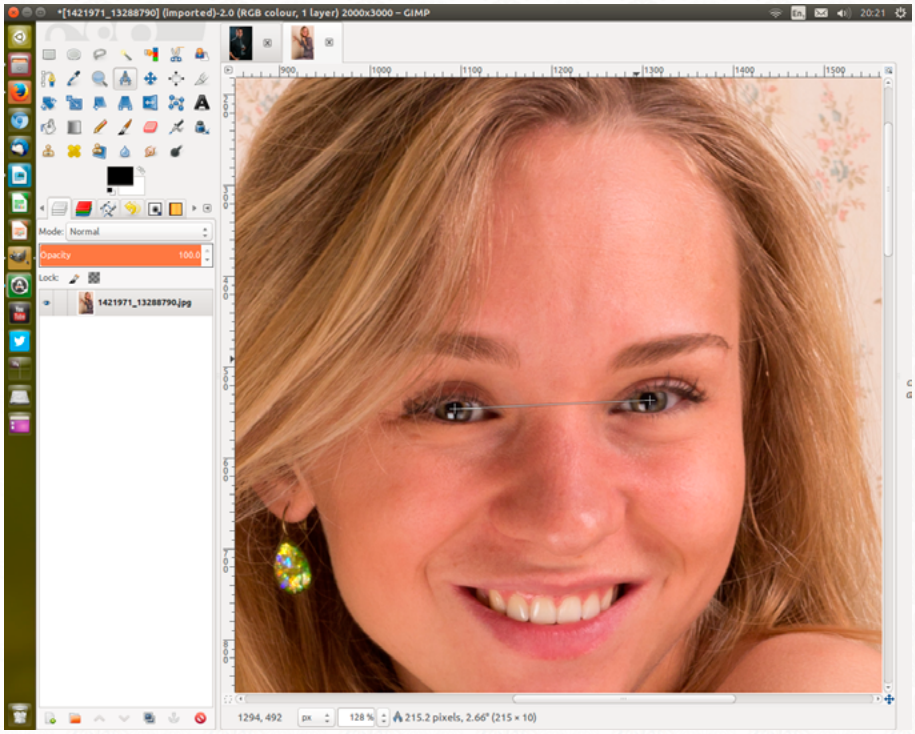
Table 1. *Details and credits*

Image	URL	Details and credits
Female model	http://www.freeimages.com/photo/1421971	Female model by african_fi
Male model	http://www.freeimages.com/photo/1348191	Self portrait by rubenshito

Step 2
Open the picture of the male model as a layer and zoom in around the eyes. Using the measuring tool, take a note of the distance in pixels between the pupils of the eyes. This should be approximately 294.7px [Screenshot 1].

Step 3

Open the picture of the female model. As the lady is looking to the left and the man to the right, flip the image by performing Image → Transform → Flip Horizontally. Repeat Step 1 with the lady's eyes; this measures approximately 215.2px [Screenshot 2].

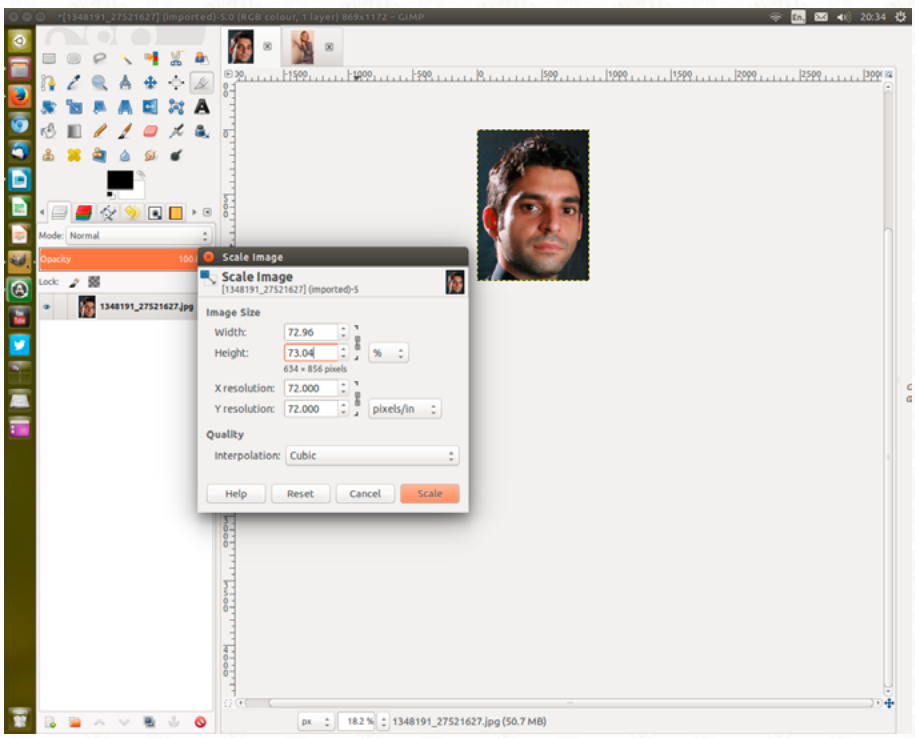


Step 4

As we will want to keep the face in proportion, we can calculate the percentage we want to shrink the male face as follows – 215.2/294.7*100 which works out at 73%.

Step 5

Go back to the male model and use the crop tool to select the head area. Scale the image using the scale tool or via Image → Scale image. Ensure both horizontal and vertical are locked. Scale the image by 73.00 percent. Press Ctrl A then Ctrl C to select all of the image and to copy it. [Screenshot 3].



Among clouds Performance and Reliability is critical

Download syslog-ng Premium Edition product evaluation here

Attend to a free logging tech webinar here

BalaBit IT Security

www.balabit.com

syslog-ng log server

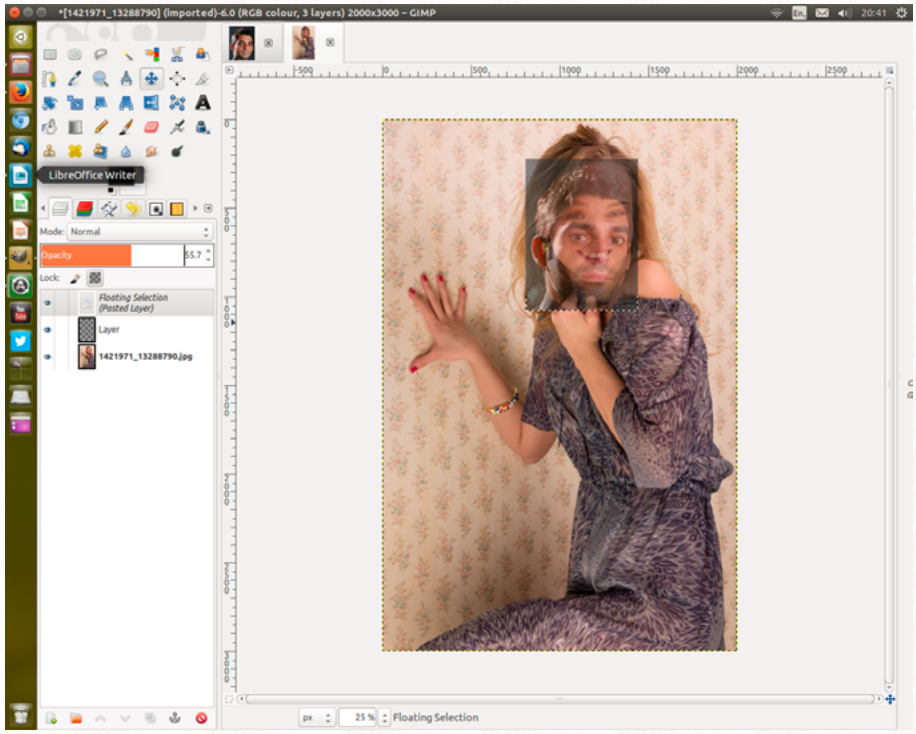
The world's first High-Speed Reliable Logging™ technology

HIGH-SPEED RELIABLE LOGGING

- above 500 000 messages per second
- zero message loss due to the Reliable Log Transfer Protocol™
- trusted log transfer and storage

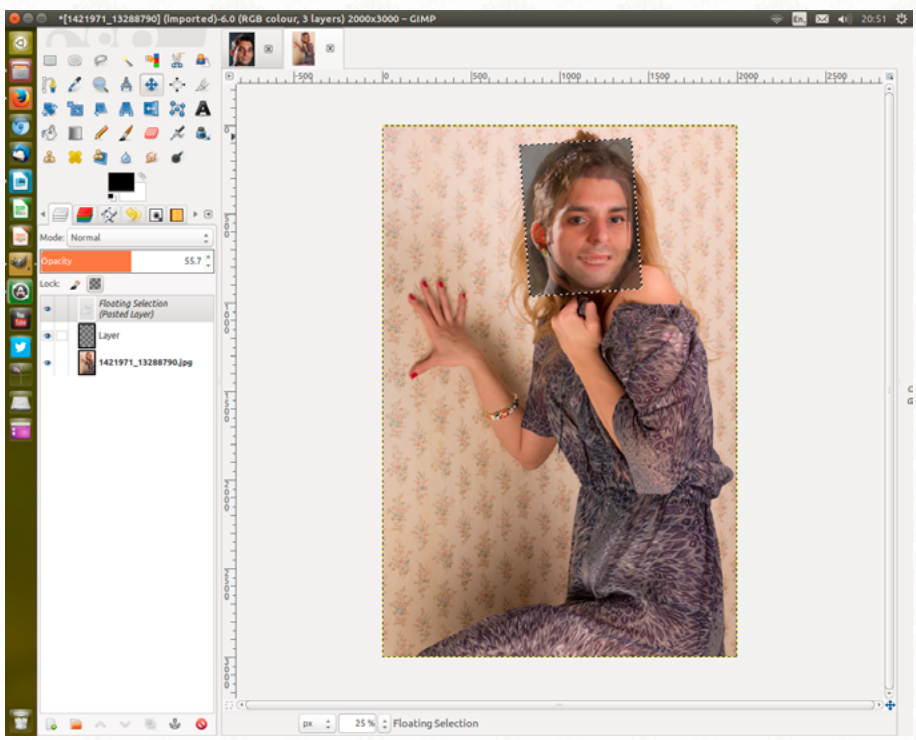
Step 6

Return to the female image and create a new transparent layer. Paste the male face into the image (Ctrl V) and reduce the opacity to 55%. Notice how both the male and female eyes are roughly the same distance apart [Screenshot 4].



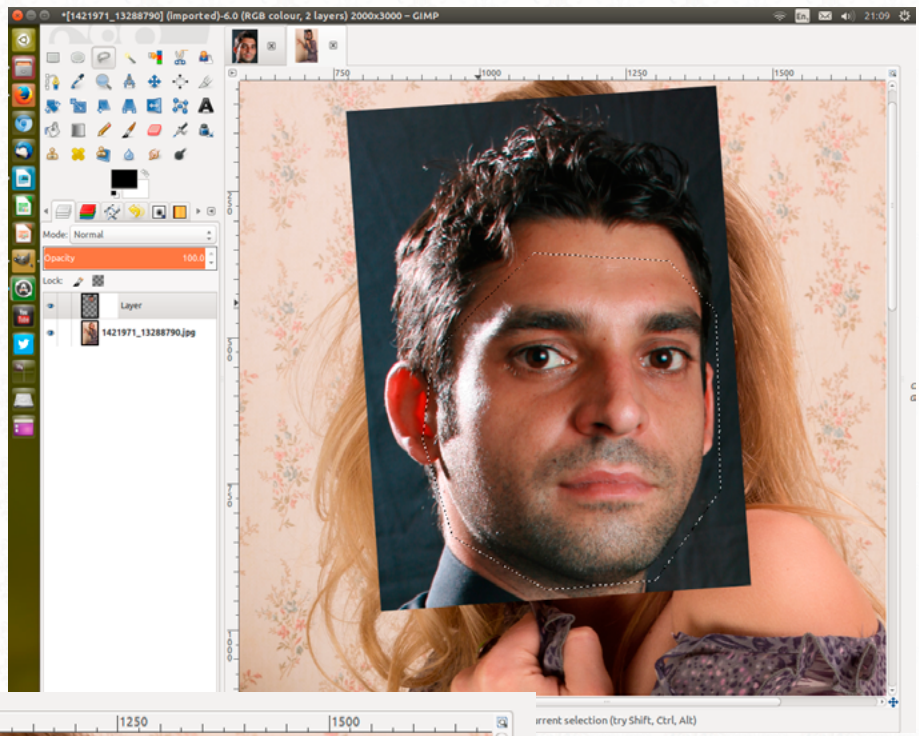
Step 7

As the lady's head is tilted slightly, we will need to rotate the male face slightly. Using the rotate tool, select the layer and move the centre point of the rotation tool to the bottom left hand corner of the image. Rotate the image by -4.0px (You may have to adjust this slightly) and, using the move tool, align the new features to match the eyes and mouth area of the lower layer [Screenshot 5].



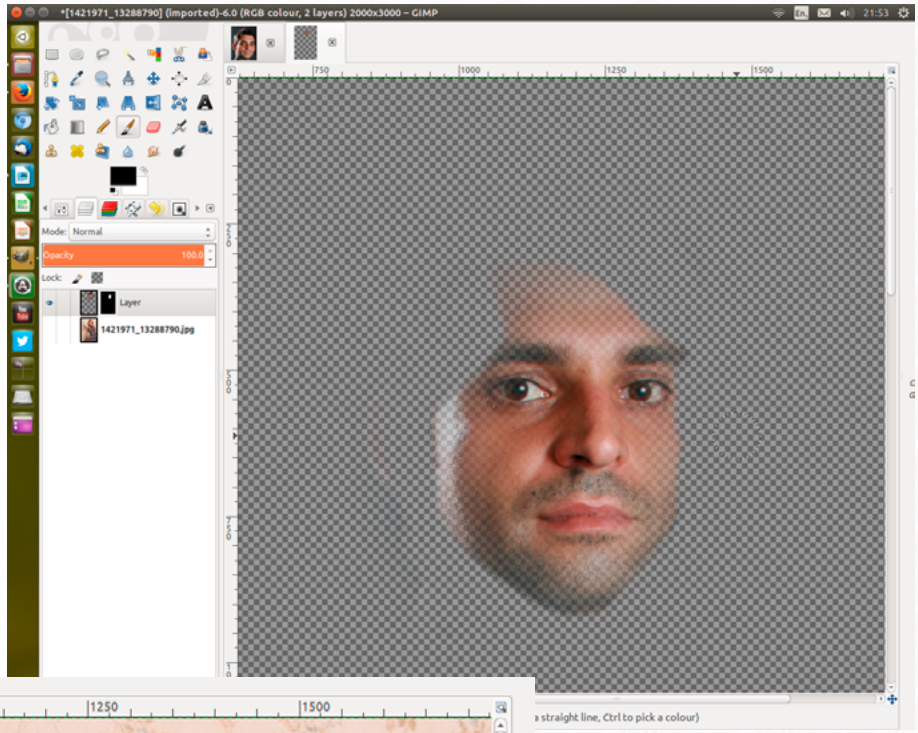
Step 8

When you are happy with the positioning, increase the opacity to 100% and anchor the layer. Right click on the top layer and add a white layer mask. Roughly select the flesh coloured area you want to transplant, Press Ctrl I to invert the selection, click on the white mask icon next to the layer preview and press Ctrl, to fill with the foreground colour [Screenshot 6 – 7].



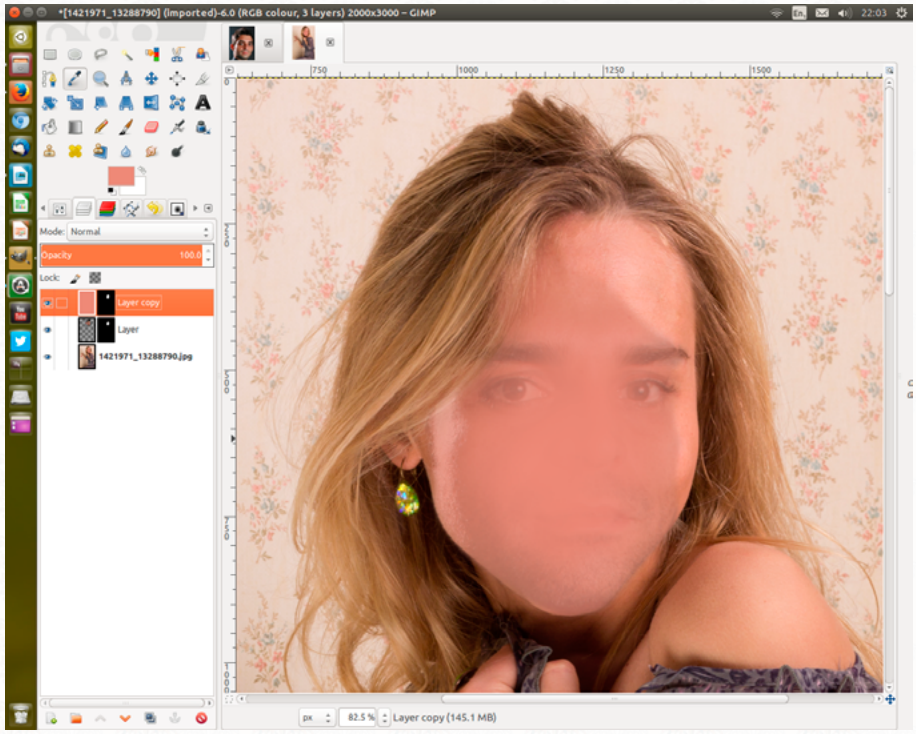
Step 9

Press Ctrl Shift A to deselect the area and, using the paint tool, remove the area of the layer you do not want included. Adjust the shape and size of the brush as required and decrease the opacity of the top layer temporarily so you can see the background. Select Filters → Blur → Gaussian Blur and apply a blur of 250px. This will increase the size of the mask, so go over the areas again with the paintbrush until you are happy with the result and no orphan areas are visible [Screenshot 8 – 9].



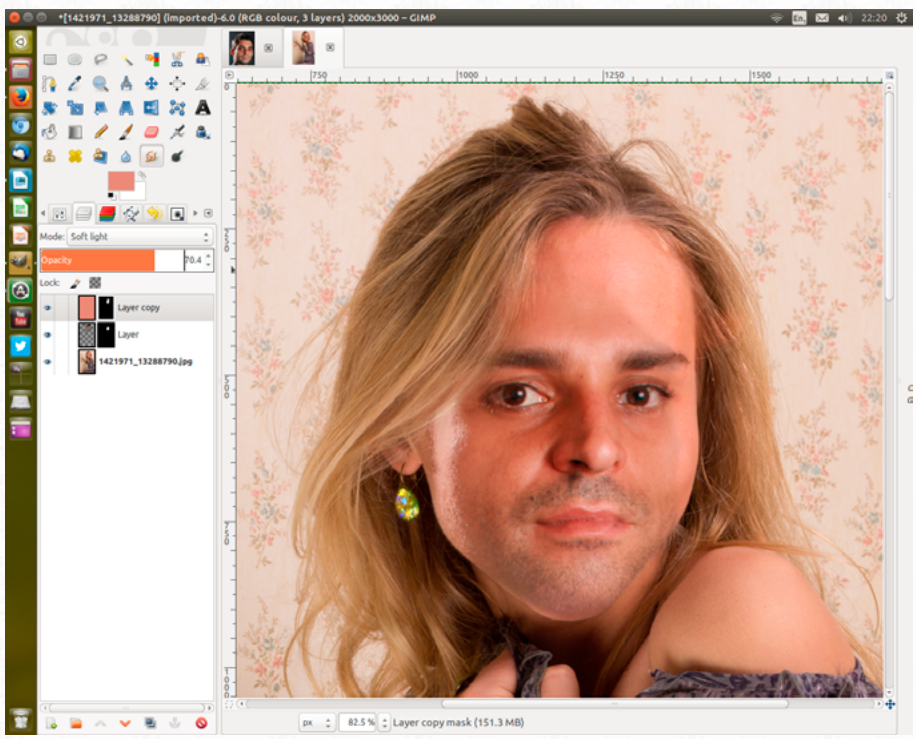
Step 10

Duplicate the top layer and make both upper layers invisible by clicking on the eye icon. Select the bottom layer and, using the colour picker tool, select a colour from just left of the lady's nose. Turn the other upper layers back on, and select the top layer icon (not the mask). Fill with the foreground colour by pressing Ctrl, [Screenshot 10].

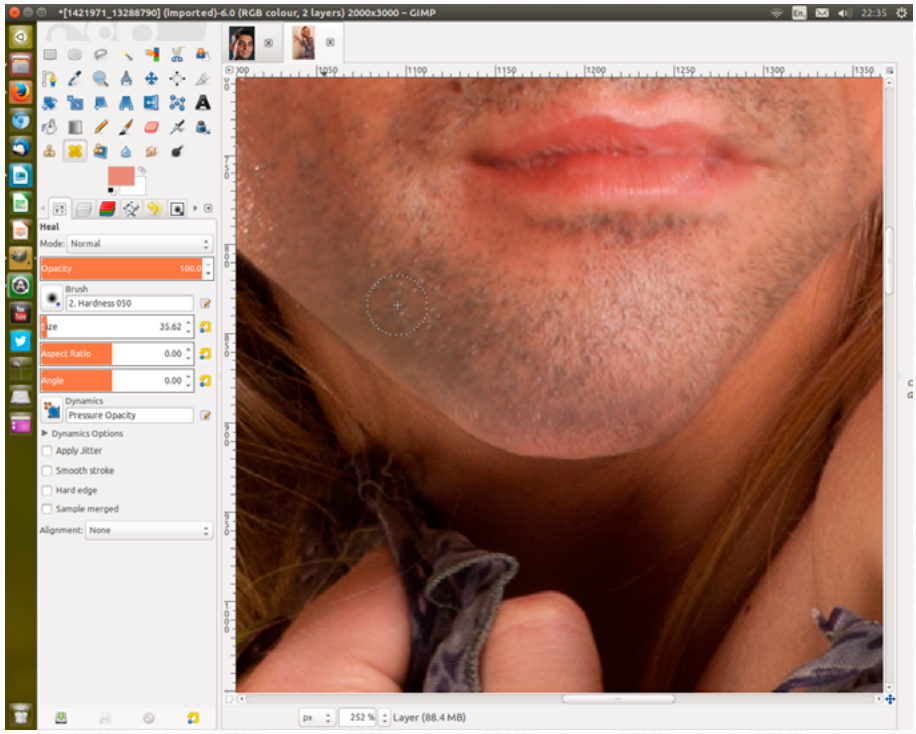


Step 11

Change the mode of the top layer to Soft light and adjust the opacity until you are happy with the skin tone [Screenshot 11].

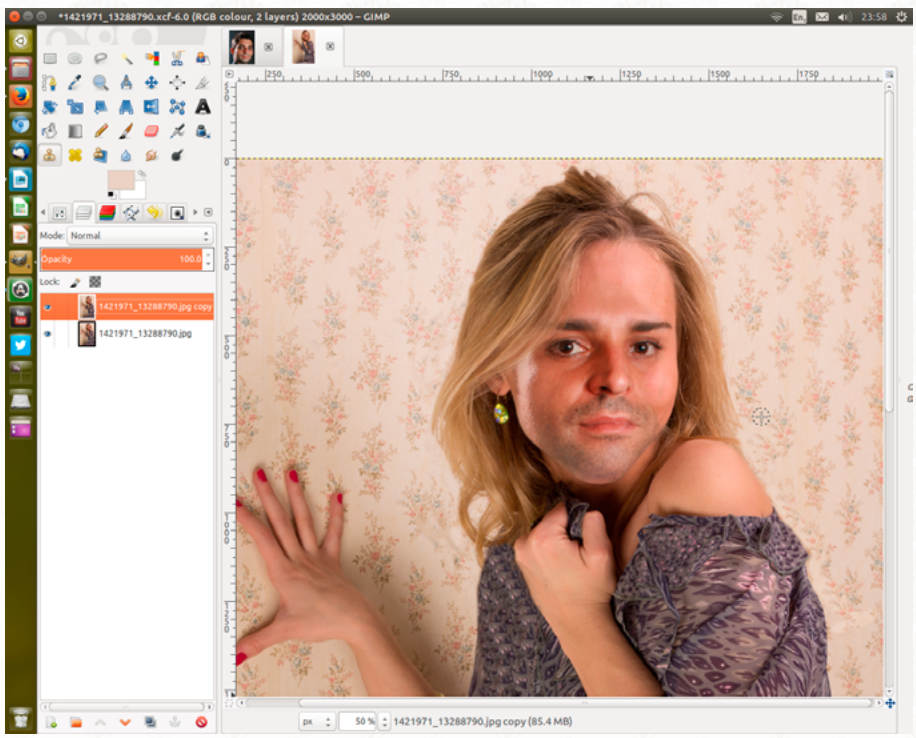


Step 12
Hide the bottom layer and right click to merge visible layers. Re-enable the bottom layer, select the top layer and, using the erase tool with 60% opacity, tidy up the area around the right eyebrow and right eye. Using the clone tool, click on an area of stubble on the chin and repair any areas around the chin that do not have stubble. If need be, move the upper layer or scale so that the face looks natural. [Screenshot 12].



Part 2 – Change the hair colour

Step 13
Merge the two layers and duplicate the result. Select the top layer. Use the clone and repair tool to remove any fine hair that we will not be able to easily select as it is too fine (e.g. around the right shoulder) [Screenshot 13].



Meet the **Developer-Friendly** Payment Solution

3 easy steps to optimized checkouts:

1

Create the checkout page

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

2

Test and optimize

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

3

Accept payments worldwide

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

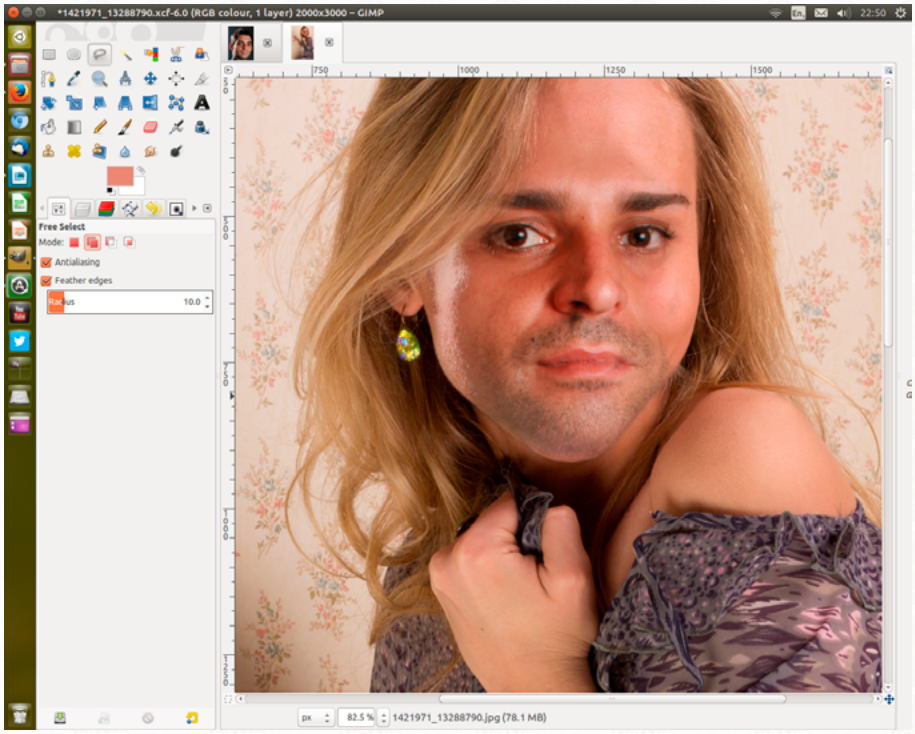
✔ Easy integration ✔ Cross-platform ✔ Secure

G2S gate2shop
Sell. More.

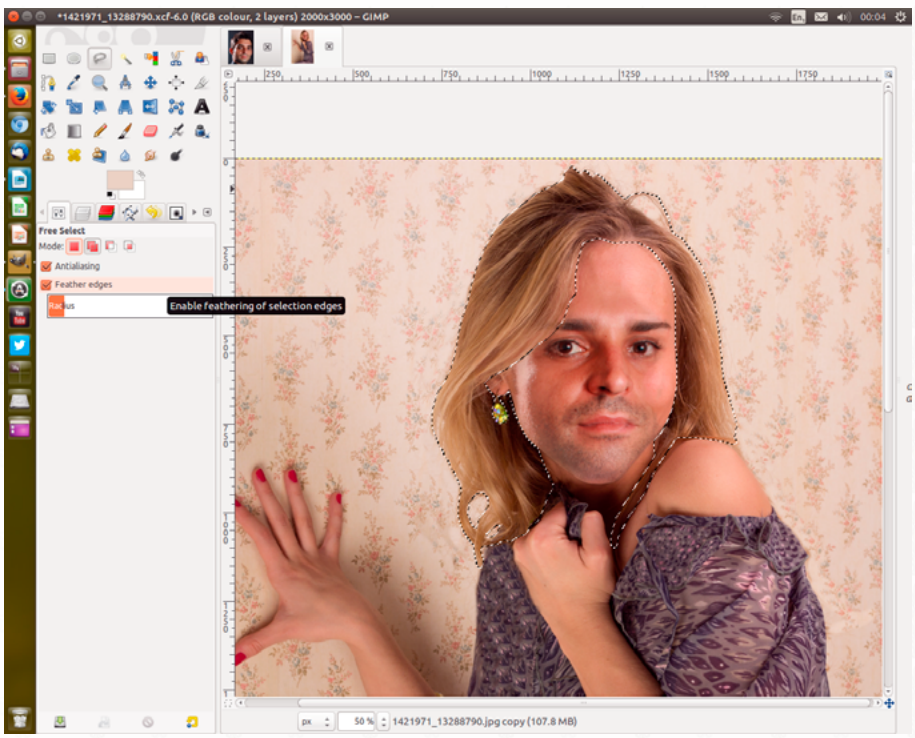
Call for a free consultation: +44 20 3051 0330

www.g2s.com

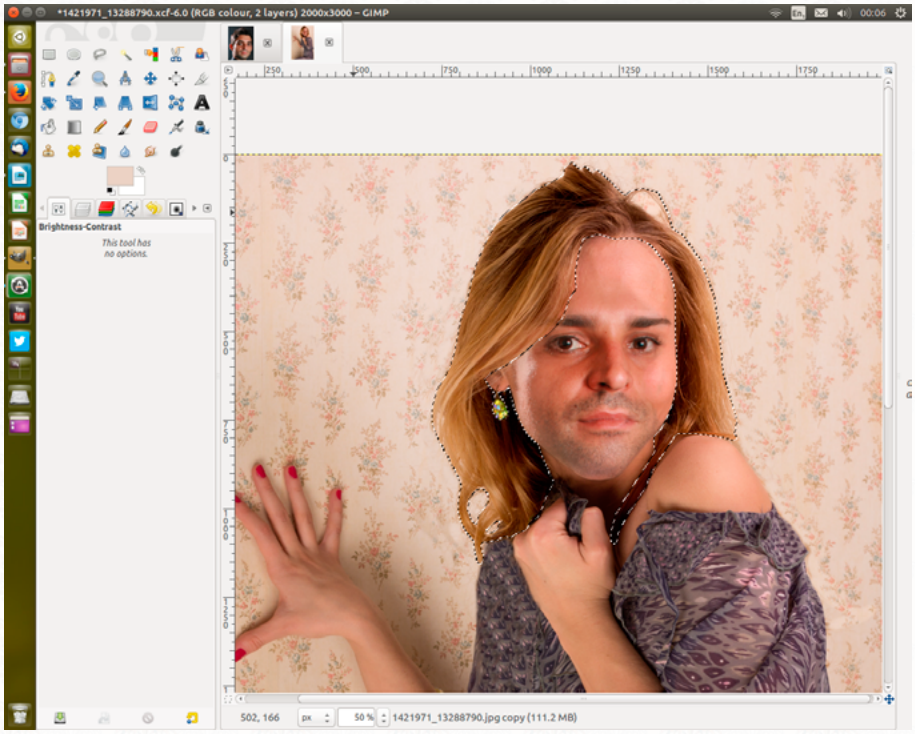
Step 14
Choose the free select tool with feathering and anti-aliasing enabled with add to current selection [Screenshot 14].



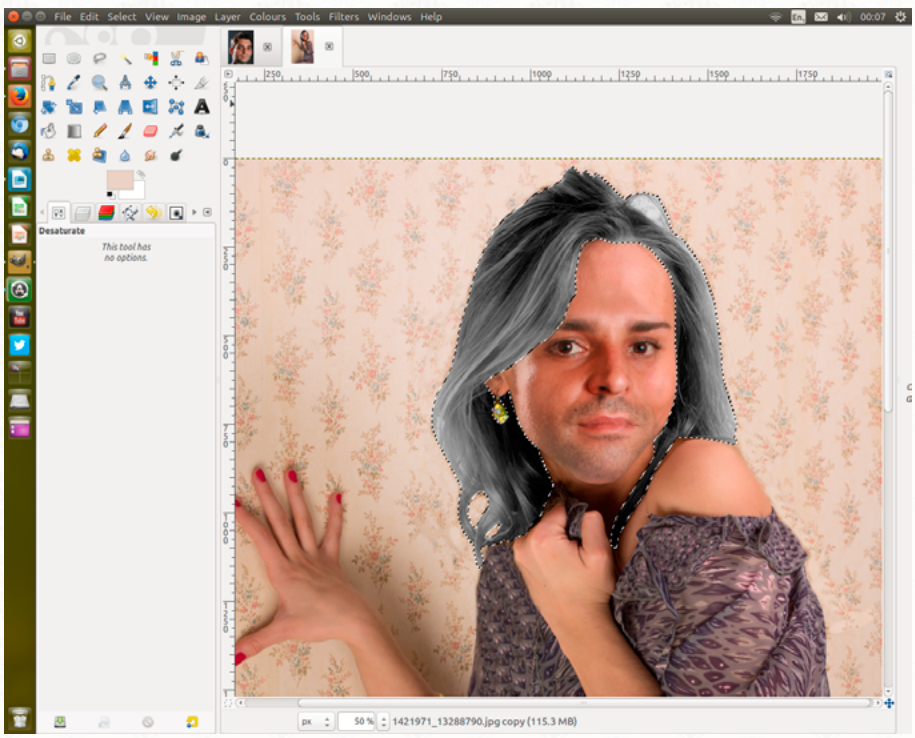
Step 15
Select the hair using the Ctrl and Shift keys to add separate areas or remove parts of the wallpaper. The more time you spend on this, the better the end result. [Screenshot 15].



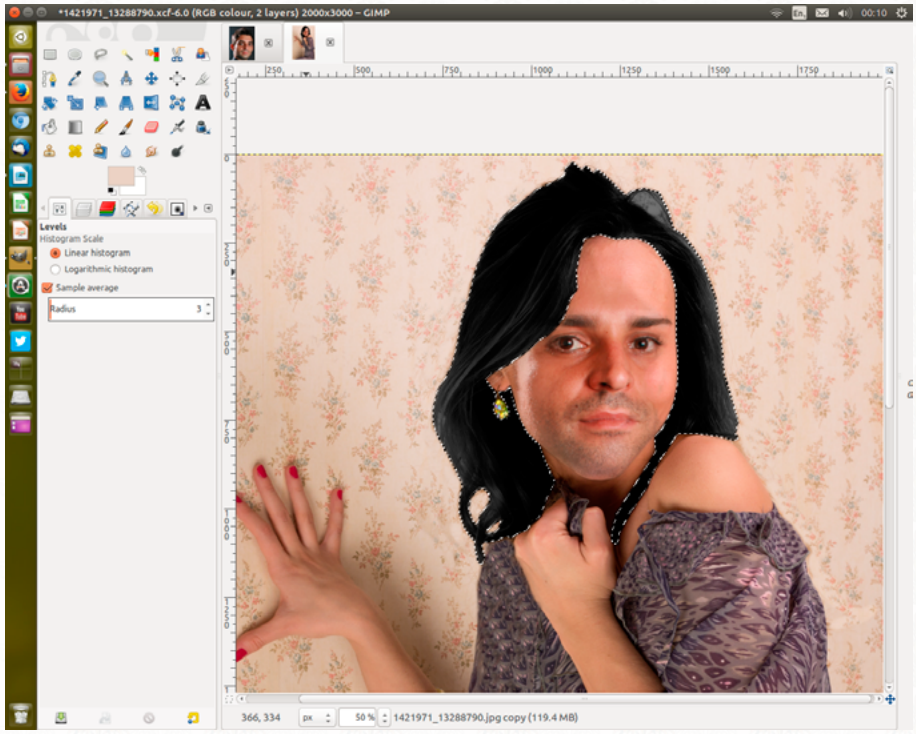
Step 16
Select Colours → Brightness – Contrast and decrease the brightness to -20 and increase the contrast to 20 [Screenshot 16].



Step 17
Select Colours → Desaturate Luminosity [Screenshot 17].



Step 18
Select Colours → Levels and adjust the input and output levels until you have the desired effect. Too much adjustment will lose contrast so we want to keep the highlights as much as possible. Adjust the brightness and contrast as required. [Screenshot 18].



Step 19
Finally, add a new layer and, using the paintbrush with the black colour, touch up the area around the right hand side of the neck with 50% transparency. Adjust the opacity of the middle layer to get a realistic effect [Screenshot 19].

While the resulting demo in this article isn't perfect, time and perseverance will improve it!

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.

Faster.
Better.
Reliable.

Trusted by over 500 ISPs worldwide.

Hyper is the first multimedia cache fully developed in Brazil, by Taghos.
With Hyper, ISPs can save on network bandwidth while increasing content-delivery speeds, resulting in end-customer satisfaction.

Features:

- 24x7x365 always-on support
- Active monitoring
- Automatic updates
- Appliance or license
- Easy deployment
- Configuration and reports via web interface

hyper

Remote Install

Using your hardware

Model	Traffic	RAM	Cache	SSD
T15	Up to 15 Mbps	8 GB	1x 1 TB	-
T50	Up to 50 Mbps	8 GB	2x 1 TB	-
T100	Up to 100 Mbps	8 GB	2x 1 TB	1x 160 GB
T150	Up to 150Mbps	16 GB	3x 2 TB	1x 160 GB
T300	Up to 300 Mbps	16 GB	5x 2 TB	1x 240 GB
T500	Up to 500 Mbps	32 GB	7x 2 TB	1x 480 GB
T1000	Up to 1 Gbps	64 GB	10x 1 TB	1x 480 GB
T2000	Up to 2 Gbps	96 GB	24x 1 TB	3x 480 GB
T3000	Up to 3 Gbps	128 GB	32x 1 TB	5x 480 GB

Visit us at **www.taghos.com** and start saving bandwidth today!

Saving Time and Headaches Using

the Robot Framework for Testing

As a developer you usually start by designing your solution, then you start coding, then you start creating your test cases and turn it over to QA, and that is all. If you did it right, there is no rework needed to be done and all is well. Then your new assignment comes, and you again start designing your test cases, but this takes time.

What you will learn…

- Familiarize yourself with Robot Framework
- Create small test cases in Robot Framework to test software functionalities

What you should know…

- Basic programming constructs
- Basic shell programming background

Maybe you test only the new functionality and ship it, but somewhere along the way you caused a regression because there was an unknown bug that your new code made appear, but now will you take the heat for it? If you value your time, you should start automating your testing procedures. You will save time in the long run. First, you must take the time to build your library of test cases but after you have completed it, it will save you time and headaches in the future.

The Robot framework is an open source tool that can help you to automate testing procedures. You could create a series of test cases to test every functionality of your software and receive a report in HTML format that will let you know if your software meets the acceptance criteria you have established or if it needs some more work, or whether there is an issue for some kind of data. It's all based on what you decide to test using a scripting language that has the concept of keywords. Keywords are just functions; they perform certain actions based on parameters. If the built-in functions/keywords are missing some functionality, you could add them using Python or Java. Here is where the Robot Framework comes in. This is the author's definition of what it does: "The Robot Framework is a generic test automation framework for acceptance testing and *acceptance test-driven development* (ATDD). It has easy-to-use tabular test data syntax and utilizes the keyword-driven testing approach. Its testing capabilities can be extended by test libraries implemented either with Python or Java, and users can create new keywords from existing ones using the same syntax that is used for creating test cases."

ATDD is a development methodology based on communication between the business customers, the developers, and the testers. As a developer you will have a clear understanding of what is expected of you and what and how the customer will want your product to behave.

The Robot Framework is used mostly for testing web applications, but you could use or modify it to meet your needs.

The objective of this tutorial is just to introduce the Robot Framework, see what you could get done with it, and maybe you will see the usefulness of it. With some Python or Java background, you could extend it to do whatever you need. Here are the current features (we are not going to see all of what is available):

Features

- Enables easy-to-use tabular syntax for creating test cases in a uniform way.
- Allows using keyword-driven, data-driven and behavior-driven (BDD) approaches.
- Provides ability to create reusable higher-level keywords from the existing keywords.
- Provides easy-to-read reports and logs in HTML format.
- Is platform and application independent.
- The modular architecture supports creating tests even for applications with several diverse interfaces.
- Provides a simple library API for creating customized test libraries.
- Provides a command line interface and XML based outputs for integration into existing build infrastructure (continuous integration systems).
- Provides support for Selenium for web testing, Java GUI testing, running processes, Telnet, SSH, and so on.
- Remote library interface enables distributed testing and implementing test libraries in any programming language. It provides tagging to categorize and select test cases to be executed.
- It has built-in support for variables, practical particularly for testing in different environments.

Requirements

I'm using FreeBSD 11.0-CURRENT r264459, but the examples will work where you have the following versions of these packages. Using pkg(1) we need to install the following packages:

- py27-robotframework-2.8.4
- py27-robotframework-selenium2library-1.5.0

Just these commands will do:

- pkg install py27-robotframework-selenium2library-1.5.0
- pkg install py27-robotframework-2.8.4

After pkg works, it's magic. We are ready to create some tests.

First, when we start creating tests, there are two ways to organize your tests. You can either create a Test suite as a text file (a Test suite is just all your tests grouped under one topic of your choice), which will contain all your created tests for a particular functionality or, as I prefer, as a directory where inside you will store all the tests related to your Test suite in whatever fashion meets your needs or likings. For this tutorial I have used the following directory structure:

```
cneira@:~ % tree RF
RF
`-- TestSuites
 `-- pkgtests
```

As a first approach to Robot Framework, we will create our first test. We will examine a couple of pkg(1) functionalities (just to give you an idea of what you can accomplish):

- Search
- Info
- Version

We want to make sure these commands work as intended, so just make it simple. Let's check if the search command works as intended. Here is our first script.

```
***Settings***
Library  OperatingSystem
Library  Process
Library  Collections

*** Variables ***
@{pkglist}  emacs  vim  xorg  tree  xfce  nothing

*** Keywords ***
Package Search
       [Arguments]   ${package}
       ${result}=  Run Process  pkg  search  ${package}
        Log  ${result.stdout}
       [Return]  ${result}

***Testcases***

Testing PKG(1) Search
       :FOR  ${element}  IN  @{pkglist}
       \  Log  ${element}
       \  Package Search  ${element}
```

Figure 1. *First example robot script*

It does not do much, but here you will see some key concepts for creating a script to be executed by the Robot Framework.

The script is organized in four sections:

Settings

Here is where you include all the libraries we are going to use. As all libraries do, they will provide us with new methods/functions to be used. Here in the Robot Framework, functions are all called *Keywords*.

You include a library using the Library statement.

Variables

Here, you declare your variables that will be global to all test cases in this file. You have available lists, dictionaries, scalar; all the types available are in this document: *https://code.google.com/p/robotframework/source/browse/doc/userguide/src/CreatingTestData/Variables.txt?r=2121dfb53e91631fd80130af95f4f36ade2ded3c*. Variable notation is the following:

```
${var} for scalar variables
@{var} is for list notation.
```

```
:FOR  ${element}  IN  @{pkglist}
\  Log  ${element}
\  Package Search  ${element}
```

Figure 2. *FOR keyword syntax*

Keywords

It is the core of the Robot Framework that all actions are performed by keywords. You can use the term "function" when referring to the keyword. It is just the same concept, you pass parameters to it to perform their magic. If there is no keyword to do what you need, you could create a new keyword using the existing keyword as we have done here. If you are out of luck and the existing keywords cannot be used as building blocks to create the functionality you need, then you still have the option to code in plain Java or Python to add a new functionality.

Here we used the Run process keyword from the Process library, taken from the documentation:

"Runs a process and waits for it to complete. Command and *arguments specify the command to execute, and arguments passed to it."

```
cneira@:~/RF/TestSuites/pkgtests % pybot pkgsearch.txt
==============================================================================
Pkgsearch
==============================================================================
Testing PKG(1) Search                                                 | PASS |
------------------------------------------------------------------------------
Pkgsearch                                                             | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed
==============================================================================
Output:  /usr/home/cneira/RF/TestSuites/pkgtests/output.xml
Log:     /usr/home/cneira/RF/TestSuites/pkgtests/log.html
Report:  /usr/home/cneira/RF/TestSuites/pkgtests/report.html
cneira@:~/RF/TestSuites/pkgtests %
```

Figure 3. *PKGSearch*

Pkgsearch Test Report

Generated
20140607 14:39:28 GMT 00:00
7 hours 48 minutes ago

Summary Information

Status:	All tests passed
Start Time:	20140607 10:39:24.286
End Time:	20140607 10:39:28.225
Elapsed Time:	00:00:03.939
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:04	
All Tests	1	1	0	00:00:04	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Pkgsearch	1	1	0	00:00:04	

Test Details

Totals | Tags | Suites | Search

Type: ○ Critical Tests ○ All Tests

Figure 4. *Our script output result report*

Test Cases

What we are going to test here is the name of our test case stated as explicitly as possible. You can write what you want, here is the clarity on what we are doing, don't be sparse with words, in this case the pkg search command. Here we just iterate for every part of the list `@{pkglist}`.

Test Statistics

REPORT

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:04	
All Tests	1	1	0	00:00:04	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Pkgsearch	1	1	0	00:00:04	

Test Execution Log

TEST SUITE: Pkgsearch — Expand All

Full Name: Pkgsearch
Source: /usr/home/cneira/RF/TestSuites/pkgtests/pkgsearch.txt
Start / End / Elapsed: 20140607 10:39:24.286 / 20140607 10:39:28.225 / 00:00:03.939
Status: 1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed

TEST CASE: Testing PKG(1) Search — Expand All

Full Name: Pkgsearch.Testing PKG(1) Search
Start / End / Elapsed: 20140607 10:39:24.420 / 20140607 10:39:28.220 / 00:00:03.800
Status: PASS (critical)

FOR: ${element} IN [@{pkglist}] — Expand All

Start / End / Elapsed: 20140607 10:39:24.423 / 20140607 10:39:28.216 / 00:00:03.793

VAR: ${element} = emacs — Expand All

VAR: ${element} = vim — Expand All

VAR: ${element} = xorg — Expand All

Figure 5. *The log functionality of Robot Framework display all the steps and results from our script*

```
Library  OperatingSystem
Library  Process
Library  Collections

*** Variables ***
@{pkglist}  emacs  vim  xorg  tree  xfce  nothing

*** Keywords ***
Package Search
        [Arguments]   ${package}
        ${result}=  Run Process  pkg  search  ${package}
         Log  ${result.stdout}
        [Return]  ${result.rc}

***Testcases***

Testing PKG(1) Search
        :FOR  ${element}  IN  @{pkglist}
        \  Log  ${element}
        \  ${rc} =  Package Search  ${element}
        \  Should Be Equal   ${rc}  ${0}
```

Figure 6. *Defining testcase syntax*

FOR statements are continued by /, so it is clear where the FOR statement goes in the code. We called our new created keyword and gave it as argument one element of the `@{pkglist}` list (Figure 2). The Log keyword is useful to see the contents of the variables in the report file.

Remember that to create our script we are using TSV (tab-separated values), so each space entered counts. Keywords can only have one space between each word, for example this keyword:

```
Fetch Clients with late payments
```

For Robot, it is one keyword with one argument:

```
Fetch Clients with and its argument is late payments
```

So be careful when you are editing a robot script file, but you can also use an ide to not worry about these details like RIDE (*https://code.google.com/p/robotframework-ride/*), or you can use intelliJidea robot plugin or emacs robot-mode (*https://github.com/sakari/robot-mode*).

Now that we have installed the Robot Framework, we could complete our test case (Figure 3).

To complete our test case, we need to call pybot because our script could execute the pkg search command. The test is recorded as PASS. There is 1 critical test since we did not use the tagging feature, by default it's marked critical.

This is the report.html outputted by pybot (Figure 4).

If you want to check the steps performed and your Log statements, they all are in the log.html file (Figure 5).

In this example, we blindly tested if we could execute the command but we did not check for the return code of the pkg search command, so this test is useless. Let's fix that (Figure 6).

Here we add the built-in keyword "Should Be Equal" to compare the return value from pkg search and the expected value. If this fails, the test ends and is marked as failed (Figure 7).

Here is the report for this execution: Figure 8.

And here is the execution log, here we see where the test failed. Pkg search returned a 70 error code, but we expected a 0 return code for success. As the last element of the list is a package that does not exist we have forced the error condition (Figure 9).

It is the end of this tutorial, and this is just a simple example of what you can accomplish. It skipped over a lot of the functionality that the Robot Framework provides, but it gives you an idea of what kind of things you could do with the Robot Framework. For example maybe you want to certificate your platform, so you create a script validating

```
cneira@:~/RF/TestSuites/pkgtests % pybot pkgsearch.txt
==============================================================================
Pkgsearch
==============================================================================
Testing PKG(1) Search                                                 | FAIL |
70 != 0
------------------------------------------------------------------------------
Pkgsearch                                                             | FAIL |
1 critical test, 0 passed, 1 failed
1 test total, 0 passed, 1 failed
==============================================================================
Output:  /usr/home/cneira/RF/TestSuites/pkgtests/output.xml
Log:     /usr/home/cneira/RF/TestSuites/pkgtests/log.html
Report:  /usr/home/cneira/RF/TestSuites/pkgtests/report.html
cneira@:~/RF/TestSuites/pkgtests %
```

Figure 7. *Failed execution for this test case*

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	0	1	00:00:04	
All Tests	1	0	1	00:00:04	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Pkgsearch	1	0	1	00:00:04	

Figure 8. *Report showing our failed vs passed tests*

Figure 9. *This is where the test failed*

References

- Acceptance is testing. (2014, June 06). Retrieved June 07, 2014, from *http://en.wikipedia.org/wiki/Acceptance_testing*
- BuiltIn. (n.d.). Retrieved June 07, 2014, from *http://robotframework.googlecode.com/hg/doc/libraries/BuiltIn.html?r=2.6.1#Create* List
- Opening library documentation failed. (n.d.). Retrieved June 07, 2014, from *http://robotframework.googlecode.com/hg/doc/libraries/Process.html?r=2.8.4*
- Robot Framework Quick Start Guide. (n.d.). Retrieved June 07, 2014, from *http://robotframework.googlecode.com/hg/doc/quickstart/quickstart.html#test-cases*
- Robot Framework User Guide. (n.d.). Retrieved June 07, 2014, from *http://robotframework.googlecode.com/hg/doc/userguide/RobotFrameworkUserGuide.html?r=2.8.1*

the version of the software installed or the version of the operative system you are running, testing network changes. Imagination and your Python/Java skills are the limits.

Conclusion

Time is of the essence, you could use the time that you save automating your tests on some other projects or more improvements. Yes, it will take some effort at first to create all your tests, but the scripting language is not that hard, and you could use Python/Java to cut around the corners. If you are practicing ATDD you should have already used it. Just look at it from this perspective: it's code that tests your code. How cool is that?

CARLOS ANTONIO NEIRA BUSTOS

Carlos Antonio Neira Bustos is a C, Unix and Mainframe developer. He develops in asm and does some kernel development for a living. In his free time he contributes to open source projects. Apart from that, he spends his time on testing and experimenting with his machines. What gives him great pleasure is solving old problems with new ideas. You may reach him at: cneirabustos@gmail.com.

Interview with Siju Oommen George

Please introduce yourself to our readers.

I am Siju Oommen George. I work as a Senior Systems Administrator at HIFX Pvt Ltd. in India. I work on both OpenSource and Proprietary Software such as OpenBSD, DragonFly BSD, Mac OSX, MS Windows and so on. I also review books while they are being written and give my input to the author before the book is published. I participate in a wide number of mailing lists, especially those guiding new users. I have also contributed to the DragonFly BSD wiki and write articles on DragonFly BSD. I am in the process of writing a book on DragonFly BSD and, God willing, it should be published in a year. I am married and have a daughter over two years old.

Could you tell us more about your background?

I had my schooling in a military school in India and was selected to join the Army but preferred to stick with my Engineering education which I could not complete due to undetected Hashimotto's disease and constant bouts with Bipolar Disorder. I joined the current company 13 years back while it was a startup with just 4 employees including me. I learned system administration from there and gradually replaced MS Windows with the BSDs, Mac OS X and Linux.

Please tell us about your proudest achievements?

My proudest achievements were replacing MS Windows 2000 Server firewall with OpenBSD 3.5 and replacing Linux backup Servers with OpenBSD and thereafter with DragonFly BSD and the Hammer file system. I also got the opportunity to train many system admins who work in several companies. I also used Hammer filesystem snapshots served through Samba for developers to have a backup available every 5 minutes as drag and drop.

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

Well, when I started using the BSDs 13 years back, BSD was still a well-kept secret. Now the scenario has changed,

and BSD is being used by more and more people. I think the day is not far when FreeBSD/DragonFly BSD will supplant Linux web servers, OpenBSD with its PF and carp implementation will supplant Linux with IPTables and PC-BSD will take much of the user share from Ubuntu Linux.

Could you tell more what the best capabilities of *BSD are from your point of view?

Well, each BSD focuses on different capabilities. OpenBSD focuses on security, and FreeBSD has a performance advantage with more software in its ports tree and supports ZFS. DragonFly BSD is focused on becoming a cluster OS with single sign on capabilities. NetBSD focuses on supporting as many platforms as possible. PC-BSD focuses on becoming the best desktop OS. Any day any time BSDs are more robust and secure than other OSes in the market, free or non-free.

What was/is your best tool to work?

Well, I love tmux and vim. But I also love working on OpenBSD PF & Squid and DragonFly BSD Hammer.

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

The best advice is that they should go through the handbooks, FAQs, and man pages. They are much better than that of Linux. Also make use of the mailing lists even though they are not always as forums. Using BSD OS helps you to learn more about operating systems and how programs interact because there is no wizard for everything. Most of the options are enabled/disabled by simply editing files. Once you get the hang of it it is very easy and addictive.

What are you looking for in terms of career development?

In terms of career development, I want to develop security products based on OpenBSD like firewalls in small boxes. One such open source software is pfSense, but it runs on FreeBSD. Also, I want to develop SAN boxes based on DragonFly BSD. FreeNAS is one such open source software but is also based on FreeBSD.

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

I would like to learn a programming language that will help me create visualisation software. Especially security visualisation software. Two of such software I like are GD Map and glTail.rb

A recent poster on http://www.theregister.co.uk lamented that with all the recent security failures and the increase of patching, IT is not fun any more. Are we facing a new dark period in the technology sector?

I am a grey-beard when it comes to IT – both literally and metaphorically. Having cut my teeth on the IBM XT, I have seen just about every cycle that hits the technology sector, from euphoric optimism to the sky is falling. I have witnessed IT slide from a hallowed profession in the eyes of management to "just another resource" and the widespread adoption of technology in areas of our lives we would have never envisaged 10 or 20 years ago. Trends and cycles come and go, but there is something different in the wind this time around.

Fortune 500 companies are bringing resources back in house, as major employers face a quality and recruitment crisis. Globalisation forced down salaries, and the opportunistic bean-counters – who appreciated the cost of everything and the value of nothing – fed the management culture with the mantra "We can cut costs and improve productivity". *Et voila*, we have the current scenario where employers are desperate for staff with enthusiasm, who think outside the box and are willing to go that extra mile to solve complex problems. I have been persistently canvassed by a close friend who works in the Nuclear sector to join his organization on the basis that they cannot recruit staff who are willing or able to think outside the artificial standard that HR requires – e.g. a degree. Sometimes, metrics just don't work.

Like all specialized areas, you can tell a skilled practitioner not by the paperwork, or the clothes, nor by age or gender but by the attitude. How they think. The way they see problems, what they see as important as values and what is ephemeral. Unfortunately the culture in the West has gone for the "Measurement and control" route, rather than a more creative solution. Don't get me wrong – discipline and management are essential – especially when it comes to large organizations and systems, but the culture has swung too much towards the fascist, and the chickens are coming home to roost. All I hear in the media (certainly in the UK) is that we have a skills shortage, but it is not surprising while we have a scenario where commitment and true value are not rewarded over paperwork, audits and internal politics and culture.

My employer has recently recruited a new project manager, who for the first time in the long queue of his predecessors, has actually commented on my performance. To some, his comments would be offensive, but to me personally, I take it as a badge of honor. To quote the immortal words of Tommy Cooper, the apocryphal British comedian and magician, "Let me tell you a story".

I was once employed by a famous British museum, and my role was to design and maintain electronic exhibits. I loved my job, and I would still be there today if the salary was enough to keep a family, but reality being what it was in the 1980's it was touch and go even as a bachelor. That said, one day a department had a serious problem with their CP/M based system and I was asked to have a look at it. Discovering that it had a hard disk failure, I informed the manager concerned what they needed to inform the supplier, and in the process saved them engineering time diagnosing the problem and hopefully shaved some money off the final repair bill. Sadly, this was not welcomed, as my head of department had a pathological hatred of the department I assisted. I was consequently marked down on my annual report as "Too enthusiastic for the job".

So lets get back to our PM, and for the sake of this article, let's call him Dave. Dave has all the credentials, and as PM's go he is a decent sort – I actually have time for the guy. He even mentioned the other day that (to paraphrase) "He will need to channel my enthusiasm". Now, taking into account the natural animosity that resides between IT and PM's in general to either default to a) Bow down to their methodologies while paying lip service and let them find out the hard way – generally just before forced change of employment or b) Ignoring them entirely, I decided to engage. I could have sat him in front of a BSD server and asked him to manage it, or asked him

to solve the particularly tricky stability solution we have with our web-server. But I had compassion. I talked about the organizational culture, and how I would not do his job for all the tea in china.

Simply put, Dave understood the battle that all IT departments and their staff have – the problem of breaking the cycle of disconnect. Ironically, like HR, IT are now relegated to a process and function, something that can be measured by performance, reliability and uptime. Unfortunately, hackers and the dark side do not respect these metrics and will do everything to exploit this disconnect on every level possible. Some would say the current system works in their favor. It takes a lot more than organization to run an organization. Synergy, commitment and (dare I say it, Dave) enthusiasm are vital.

I'll be the first to admit, I am like a golden retriever – throw the ball and I'll fetch. I don't have time for politics, petty squabbles, or inter-departmental rivalries. I want to demonstrate my skills, bring "magic" to the end user, and ultimately the customer. I am goal orientated. Yet, for so long, IT has not been allowed to manage itself, and have accountability where it matters – at the board level. Yet, as the heart of any organization, like Dave, we are deeply misunderstood.

So let's put this scenario in perspective. A major communications company spends obscene amounts of money relocating staff on a regular basis so that they can have a window seat. I doubt if Dave would recommend this, but it is a rather good analogy as to why we are in the mess we are in, and why it will get worse. Instead of confronting the elephant in the room, we spray deodorant, place proximity sensors around the perimeter, and from a cultural perspective deny the existence of the aforementioned mammal.

From a security perspective, all bets are off. We have moved from a battlefield of munitions to software, vulnerabilities, and Intel. The first line of defense will be the firewall, and those that support and man it. With management's track record over the past 20 years will we have an army of amazons to repel all boarders or those that have had all the stuffing systemically knocked out of them? Enthusiasm and fun aside, there is no test for a hero.

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.

UPDATE
NOWWITH
STIG
AUDITING

"IN SOME CASES
nipper studio
HAS VIRTUALLY
REMOVED
the NEED FOR a
MANUAL AUDIT"

CISCO SYSTEMS INC.

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

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

2012 Computing Security Awards
WINNER
Enterprise Security Solution of the Year

Security Products' GOVIES
2013 Government Security Awards

2012 Computing Security Awards
WINNER
Network Security Solution of the Year

www.titania.com

Great Specials

On FreeBSD® & PC-BSD® Merchandise

Give us a call & ask about our
SOFTWARE BUNDLES
1.925.240.6652

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

$29.95
PC-BSD 9.1 DVD

$49.95 Save a BUNDLE!
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
Users Guide FreeBSD Handbook, 3rd Edition, Admin Guide
FreeBSD 9.1 CD or DVD set
FreeBSD Toolkit DVD

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

Bundle Specials!
Save $$$

Just Plain Fun
Mousepads & Novelty Horns

BSD Magazine
Available Monthly

For even **MORE** items
visit our website today!

www.FreeBSDMall.com

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**

Net Open SERVICES

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

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

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

WWW.NETOPENSERVICES.COM • CONTACT@NETOPENSERVICES.COM