news@UK

The newsletter of FLOSS UK, the new name for the UK's oldest Open Systems User Group, UKUUG

Published electronically at `http://www.flossuk.org/Newsletter`

Volume 23, Number 1 **ISSN 0965-9412** **March 2014**

Contents

From the Secretariat

Jane Morrison

Thank you to all the members who have paid their subscription invoice promptly this year. The outstanding subscriptions will be chased at the end of March and any not paid at the end of April will not receive the June Newsletter.

At the time of writing we have everything in place for the forthcoming Spring Workshops and Conference being held in Brighton from 18th – 20th March. The event is being kindly sponsored by Google, HP, Bytemark and our Gold Sponsor member SUSE.

It has been a very busy time and the event looks as if it is going to be a great success. A description of the event will appear elsewhere in this Newsletter.

In February we held two very successful tutorial days in London: "GIT for Development Teams" by Lorna Jane Mitchell (6th February) and "Tuning PostgreSQL" (26th February) by Simon Riggs.

We are currently in talks with O'Reilly to bring you some more tutorials later in the year.

Other forthcoming events include:

- Bar Camp – June – Birmingham
- Unconference – October – London

Looking ahead to next year, the Spring 2015 Tutorials/Workshops and Conference will be held in March in York: more details will be published later.

The next Newsletter will be the June issue and the copy date is Friday 16th May.

To comment on past or future events, or if you have something to say about our User Group or this Newsletter please contact: **`newsletter@ukuug.org`**.

If you prefer not to receive future issues of the Newsletter in hard copy (all issues can be found on our web site in pdf format) please let me know.

Chairman's Report

Kimball Johnson

Events

I have just recovered from the Spring Conference in Brighton, I am happy to say it was a great success, with a great selection of talks, and a wonderful dinner. I want to to reiterate my thanks to all the sponsors, HP, Google, Bytemark and Brandwatch. Many thanks also to the association sponsors, Suse, O'Reilly and Linux Magazine.

Two of the three tracks were filmed, and the videos will be made available in due course, along with the slides from the presenters.

Congratulations to the winners of our best Speaker competition, Wim Godden as best overall speaker, and Mark Keating for best lightning talk. Honorable mentions also go to Bernd Erk and JP Mens.

We have also had a couple of tutorials recently, Git for Developers by Lorna Jane Mitchell and PostgreSQL Tuning by Simon Riggs. Both of these were well received and we hope to run them again in the future, if you are interested please contact the office and we will let you know when they run again.

I would also like to encourage anyone with ideas for courses they would like to be run to email the office with their ideas.

Josette Garcia

I am sorry to pass on the news that Josette Garcia has been made redundant from O'Reilly. She has been with them for 25 years, and for the past 18 years been heavily involved with promoting open source, and a regular sight at ours and other's conferences. I know you will all join me in wishing her well for the future.

Support for local user groups

The budget is still available for FLOSS UK to assist local user groups by helping them to obtain speakers for their events and assist with travel expenses. In addition we have a small budget for assisting with projects that would benefit the Free Software or Free Hardware communities in some way.

If you have an idea and wish some support, please contact Jane on **`office@flossuk.org`** and it will be discussed by Council.

Get Involved

FLOSS UK exists to serve its members and we are always on the lookout for people who are keen to get involved in any capacity, whether that be through volunteering to help with organising events, writing newsletter articles, or entirely new activities which haven't been tried before. If you would like to help out in any capacity please do get in touch via **`office@flossuk.org`**.

Spring Conference at Brighton

Roger Whittaker

The Spring Conference was held at Brighton this year, at the Old Ship Hotel on the sea front.

Four half day tutorials / workshops took place on Tuesday 18th March: the topics being LDAP (Andrew Findlay), Ansible (Dag Wieers and Jeroen Hoekx), Learning Perl (Ian Norton) and web caching (Wim Godden).

The conference proper was on Wednesday 19th and Thursday 20th: unfortunately a combination of work and personal commitments meant that I was only able to attend on the Wednesday. There were two tracks (three at certain times), so what follows is my description of a small subset of the talks that were offered.

The conference opened with an introduction from Kimball, followed by a keynote address by Paul Downey of the Government Digital Service. That a government speaker should be providing our conference keynote address shows what a long way we have come. Parts of government IT are of course still in a dreadful state with a top down, big supplier procurement mentality. But Paul Downey's talk showed that those providing the public facing services via the **`gov.uk`** domain really do "get it". I was most encouraged and inspired by this talk.

Howard Chu, beginning as ever with a brief violin solo, spoke on new features in openLDAP, particularly with reference to the massive performance gains that have been achieved by the switch to LMDB, a back end that has also been adopted by various other projects.

I particularly enjoyed Jan Piet Mens' clear description of the Ansible configuration management system. This sounds to me like "configuration management done right" and the talk inspired me to look into Ansible further.

Simon Riggs gave an informative "state of the union" talk on PostgrSQL, detailing the new features in 9.3 and what to look out for in the upcoming 9.4 and 9.5 releases.

Mark Keating of Shadowcat gave an amusing and informative talk on the state of Perl, based around Mark Twain's famous "reports of my death" quote. Statistics from CPAN and the Perl community showed the extent of the exaggeration. He gave various examples of important new modules that Perl users should be aware of, particularly emphasising the usefulness of `local::lib`.

Wednesday ended with lightning talks. JP Mens talked about OwnTracks: a mobile app reproducing the functionality of the now defunct Google Latitude, using MQTT to communicate with a mosquitto MQTT broker instance. I was sufficiently inspired by this to set it up as soon as I got home.

Charles Yarnold from London Hackspace introduced the (somewhat bizarre sounding – though I didn't see it) "games caravan" which entertained delegates later that evening.

Tim Fletcher talked about KVM on ARM (using an AllWinner A20 board), and Andrew Findlay gave a most entertaining short talk on his home automation project, controlling hot air heating with a system running on a Cubox device.

Mark Keating's "I do not like Perl and CPAN" (based on "Green Eggs and Ham") was wonderful. His colleague Matt Trout provided further plugs for `local::lib` and reasons not to suffer from CPANphobia.

Sadly, I had to return to London after the lightning talks. My colleague Paul Thompson writes:

> *The Conference Dinner was held at 'Alfresco' bar and restaurant, close to the remains of the ill-fated West Pier. It's a spacious, modern and smart venue, and was generally judged to be an excellent choice. Service was efficient and friendly, and we were treated to a copious, eclectic, quality buffet. There was plenty of complimentary wine too. The evening was an ideal opportunity to get to know other delegates, and lively conversation continued until very late – then it was just a short, invigorating walk along the seafront back to the conference hotel, The Old Ship, where most were staying.*

In addition I understand that the London Hackspace caravan was operating, providing fun and fascination during the evening.

Among the talks on Thursday that I very much wish I had seen are JP Mens on MQTT and David Proffitt on "Secure Automated Connections with SSH". Slides and recordings of talks are being put up at **`http://www.flossuk.org/Events/Spring2014/Talks`**.

The conference was sponsored by SUSE, Google, Bytemark, HP and O'Reilly, who had an exhibitors' area near to the conference rooms (but unfortunately not in the same place as the coffee).

Kimball has already mentioned the shock that we all felt on hearing at the conference that Josette Garcia has been made redundant from O'Reilly. This inexplicable decision robs us of a friend who was a constant feature of our events over the years and someone with a true understanding and concern for multiple open source communities across Europe.

FLOSSUK Spring Conference 2014

Mark Keating

The Thursday; the Event, the People and the Prize

It's spring in the Western Hemisphere when all us bipedal mammals who are chilled, blasted and drenched[1] by an extended Winter look forward to warmer days. In the realms of the free software movement in the United Kingdom, and European partners, we are also looking to the FLOSSUK Spring Conference. This year FLOSSUK Spring was held in the cosmopolitan seaside resort of Brighton.

This is only the second[2] FLOSSUK that I have attended and the first where I have had the fortune to be asked to speak. Shadowcat, the company I am a partner in, have attended for a number of years sending Matt S Trout and Ian Norton to give talks or help out.

This year I was asked to give a talk on the current state of the Perl world in regards to developers, culture, environment and practices. At the same time I was to have a peek at some of the more modern tools. As I am known for being a semi-prolific man of words I was also asked to write about the second day, the Thursday, of the conference event.[3]

FLOSS Spring is an event that, in popular mythology, is mostly attended by systems administrators. I feel this is a historical precedent, or maybe just a popular perception, that doesn't match the character of either the attendees or the phenomena. The talks, people and passion maybe all about modern techniques in DevOps, and a significant percentage are systems people. However they are also interested in languages, projects, development as well as systems. One of the best features is born from the fact that this conference is language agnostic. If it is open, it is wanted. And most likely respected, accepted, loved and discussed.[4]

Though a large number of the attendees may work in system engineering, they are also closely connected to development and deployment. This is perhaps the reason for the eponymous DevOps association. What we actually have is a broad sector of senior system managers, network architectural engineers, developers, system administrators and project managers. This is a great environment for cross-pollinating ideas, techniques, software solutions and problem solving.

The talks I attended were all very well thought out and presented. I felt sometimes massively out of my depth as DevOps is not an area I get to study or be in conversations about. However this is a good thing as I know I was learning from each of the speakers. Although I was aware of many of the software packages being discussed I have never previously had the chance to attend presentations from experienced practitioners discussing them in detail.

The Thursday started with an almost Open Source conference tradition. Bytemark, once again, supplied a personalised conference gift in the shape of a mug. This had a unique message, conference branding and your own name. I love these gifts and thank Bytemark for them. After the opening address I wandered into the lecture theatre and settled myself for a long stay. I was fortunate in that most of the talks I had elected to see were in the same location so I could grab a coffee and get comfortable.

The first talk I heard was given by Bernd Erk, who gave the first of his two talks on the Thursday, on OpenNebula. OpenNebula is a datacentre virtualisation system, however think more that it creates 'private clouds not public clouds'. Bernd introduced us to the general system and some of their forward plans such as the wish to move to an agreement with Amazon (AWS) for extension into the public space. This would then allow companies to create a private, or internal network, cloud, but perhaps have an interface layer, or abstraction layer that interoperates with AWS to give a common public gateway.

OpenNebula however is all about small clouds for internal company usage. A highlight is that it works with a range of open tools to manage configuration, replication and reporting.

Chris Jones, gave an interesting talk about OpenStack. The latest move in OpenStack is called TripleO. The short, and mostly true version of this, is to be running a cloud with VMs and attached storage using OpenStack on OpenStack. Hence why they are calling this TripleO, OpenStack On OpenStack you see, almost alliterative. The idea is to use the same tool to deploy as the tool that is being deployed, as many of the components to do this are already available inside that tool (and now I get lost in recursion).

The thinking for using TripleO seem fairly sensible, they want to have tools that work well within their ecosystem and to a methodology that has worked for them. This isn't, well not in principal but maybe in practice, a land-grab for tools that are currently third party. It is born out from a number of necessities that make practical and logical sense. Principally, the third party tools that are available don't always follow the API methodology and there is little upstream movement to push into the OpenStack ecosystem by the third-party content vendors.

Conceptually they see it as a cloud on top of a cloud. This is not strictly true, but it helps as a conceptual model to understand what is actually happening. It gives them the name of undercloud – the abstracted management layer that sits on the hardware; and the overcloud which is the client facing open stack implementation.

The second talk of the day given by Bernd Erk was a quite interesting discussion on the roadmap for Icinga2. Icinga is a fork of the ever-popular Nagios and is compatible with the modules from the Nagios eco-system. The current roadmap seems very robust with a lot of development plans.

It was good to hear Bernd talk with a lot of enthusiasm for the current state of network monitoring and cloud based PaaS.

Matt S Trout gave a talk on Prolog and Devops Logique – as always Matt gave a wander through the recesses of his mind, the history of computing and system approaches, with various conversations concerning configuration of servers. This wander concerned varying system architecture languages and how they helped him to imagine a development machine that allows variable configuration but can be automated. Also, it seemed, how to gain access to production machines to allow ad hoc repair without being attacked by a configuration manager.

Matt always seems to have a long term series of goals concerning: system stability; good toolsets; system automation that shouldn't constrain individual choice; the ability to make a one-off change; and the opportunity to work on unique instances.

On his journey Matt assessed the manner in which a number of other languages have implemented or addressed some of the issues he is facing. This approach allows him to evaluate and understand the argument in a much broader manner. The end goal seems to be to handle dependencies and to manage them well on a system perhaps utilising the Perl scripting language.

David Griffith gave one of the best talks of the day on Test Frameworks and the issues they have encountered and overcome at Durham University. It was strange to listen to a talk that was both complimentary on the achievements yet also modest and reflective. David was speaking about the cultural changes in remote workers and cloud systems, which nicely closed the event for me as we started with a keynote that emphasised a revolution in the culture of a department.

The event itself was closed by Kimball Johnson who presented Josette from O'Reilly with flowers as this is her last conference as an O'Reilly representative. This was a deep sadness for most of us at the conference. Josette has been a great treasure to this event and many other open source communities and she will be greatly missed as O'Reilly's voice in the community.

Kimball then presented awards for the best talk, best lightning talk and several honourable mentions. I was greatly honoured, and proud, to receive the award for best lightning talk and an honourable mention for my longer talk. The best prize however was simply being at the event and enjoying the talks and the people.

A few other thoughts have to be conveyed before I close this piece.

One of the major highlights of the week was the 'Spaceship' that was built, created and manned by the chaps at the London HackSpace. They brought this wonderful experience to the conference dinner and teams of happy victims were led into the event over the course of the night. I think I would quickly run out of synonyms if I tried to write a range of superlatives on the brilliance of this experience. The spaceship looks amazing, the various effects, screens, buttons and stories have a lot of thought and care gone into them. The LHS crew both run the simulation and act out the varying storylines with the crew and even dress for the occasion.[5]

While on the subject of superlatives of greatness allow me to think of the event as a whole. I would like to take a few moments to thank the sponsors for all their support, with an especial thanks to Bytemark for all their work, sponsorship and once again, the ace mugs. Thanks to SUSE, Google, HP and O'Reilly who exhibited over the conference days.

I would like to thank the conference organisers, and especially Kimball Johnson, for inviting me to talk at the Spring Conference and for asking me to write on my experience.

I also really liked the talk by Jans Mens on Ansible, unfortunately it quickly lost me and I think I managed to confuse his section on Roles with the ideas of Roles from Computer Languages, with particular reference to how they are used in Moose and Perl6.

I enjoyed Simon Riggs' talk on PostgresSQL. Postgres seems to be a fast-moving database system. I know a lot of effort has been expended on noSQL of recent years but Postgres has some real power and features (being able to run eighty-five million request an hour on a low powered laptop being just the thin edge of a large wedge) and a yearly development cycle that keeps a high momentum. It was also interesting that PostgresSQL suffers from the same issues that Perl does in regards to being seen as old or monolithic, I never saw it as such and so was unaware of the perception.

One of the best parts of Simon's talk was the unexpected memory issues that caused his laptop to collapse, to which Simon calmly ignored and continued presenting while an orga-tech effected a lightning fix and restart.

Notes:

[1] Though the term might be thoroughly soaked if they live in the UK.

[2] Second is by a long stretch of the imagination and refers only to the fact that I attended last year to have lunch and say hello to a few people.

[3] A tiny issue I have is that I wanted to mention the excellent talks I attended on Wednesday. The keynote speech on the changes to the UK Government website was enlightening, especially if you deal with site design and delivery. The approaches, changes and entire revolution of ideas was fascinating to learn.

[4] For that fact alone we should be shouting to all the other language groups and saying come one and all, attend, mix and learn, be a part of an open forum not a closed view.

[5] For a better description of crew see willing participants, attendees or as they are also known, victims.

Git for Development Teams Training Session (6th February 2014)

Kimball Johnson

“Git for Development Teams” was an interesting title, I mean, who else uses source control than developers of some kind?

Well maybe a better title would be Git for Non-Nerds, or Git for Normal People. Although I have been using git for some time, and struggled through its obscure syntax and ways, most of the room that Lorna had to teach were very new to git, some new to version control in general.

After a brief survey to gauge knowledge and what people wanted out of the course, what followed was an excellent breakdown of the usage of git for your everyday development tasks, staging and committing, the difference between the two, and how to get the diffs of the different parts to see what you have done. These basic techniques can be especially hard for users of other version control systems, such as subversion, as the multi-stage commit can be confusing.

Only once these basics were clear did Lorna venture into the more complex tasks of branching, merging and the collaborative distributed aspects. Still though, it was very focused, geared towards the more normal development practices, no mention of formatting patches to send via email for example; rarely used outside the Linux kernel team.

Finally once the bulk of the material was covered, Lorna went back to the feedback from the start, and discussed in quite a lot of detail the migration from subversion to git, both in the sense of how the user will change their habits, to how to actually migrate the data from one system to another. This raised the interesting point that the data is only a small part of a source control system, often there will be a myriad of hooks and scripts into continuous integration systems that need to be changed too.

Dotted throughout were tips and tricks to make life easier, right down to how to get a nice prompt, or show the useful details in log history. Overall it was a valuable course, and even myself as a seasoned git user learnt something new. I hope that FLOSS UK can run it again in the future.

PostgreSQL Training by Simon Riggs of 2ndQuadrant (26th February 2014)

Holger Kraus

On 26th February a group of students descended into a room in the basement of a hotel near Euston train station, with almost no mobile network coverage so that we could all concentrate during an intense training day about Tuning PostgreSQL For A Large Scale Web Environment.

After a short introduction about its history (PostgreSQL being a successor to Ingres), its use of the BSD licencing model and the global Postgres Community, we went straight into looking at its server architecture and server tuning. An understanding of the relationship between shared buffers, locks, procs, WAL buffers (all in memory), and the `pg_xlog`, database directory and table spaces (on disk) is crucial for successful tuning decisions. WAL stands for Write Ahead Log and is what Postgres calls its transaction logging. Interestingly, in Postgres there is no need for the traditional rollbacks found in other databases, due to the way writing data is performed: changes are first recorded in the WAL buffer, flushed to disk (in `pg_xlog`), then the shared buffers are being updated, and finally a checkpointer process writes to the database directory.

The software architecture is similar to what you find in, say, Exim: there is one postmaster process which receives all new connection requests and then forks separate background processes which deal with the queries, plus there are a number of special background processes for things like checkpointers, writers, auto-vacuum, logging and stats.

We looked at MVCC (Multi-Version Concurrency Control), VACUUM and server maintenance. MVCC means that the server will maintain multiple versions of a row of data as necessary, e.g. to ensure that short read processes can be fulfilled whilst a long write process is underway. This is the main reason why Postgres has VACUUM, i.e. literally cleaning up unused leftovers (called 'dead rows'). This is done automatically in recent versions of Postgres.

We were also delving into Locking and Concurrency, were introduced to Client and Advanced PostgreSQL Development, and found out that there is no server built-in command for data loading, from say, CSV files – instead one would use COPY or the `pg_load` utility (the slides for this were unfortunately missing from the otherwise extensive documentation we were provided with).

Postgres has a number of unusual but potentially very useful data types, such as 'arrays', 'composite types' (e.g. for storing JSON objects), 'hstore' for attribute/value pairs, and there is an initial implementation of 'materialized views'. We learnt about functions and some concepts in Postgres, such as the very useful 'window functions', the 'PL/…' Pluggable Language architecture, useful 'triggers', weird 'rules' (which you'd normally avoid using), 'asynchronous notifications', XML functions such as XPath, and '`RETURNING`' which combines `SELECT` and `UPDATE` into a single statement.

Other advanced topics covered were `EXPLAIN` and SQL Tuning, Workload Analysis, Joins and Advanced Planning, and Partitioning. However, it was stressed that with all web applications interfacing to an SQL DBMS, a lot of performance issues can be resolved by app tuning (up to x100 performance increase) rather than server tuning (perhaps x2 performance increase).

It was pointed out that the Postgres manual consists of over 15 thousand pages, all accurate and written by the developers at the time a feature is being added or updated, which is very reassuring to know.

All in all, we had an enjoyable day, packed with advanced knowledge about the Postgres landscape and advanced server use, delivered by a truly knowledgeable expert involved in Postgres development for many years. The tutorial could easily have been extended to two days, to help consolidate the items discussed.

Teaching Computing: Deadline September 2014

Les Pounder

Over the past six months, I've worked with many ICT (Information and Communication Technology) teachers who teach from key stage three (KS3) and upwards. The most common query I've had is how to prepare for the changes that are coming this September, so I thought that I would share a few of my ideas with you all.

1: How can I use the Raspberry Pi in my lessons?

When the Raspberry Pi was announced, it was heralded as the beginning of a new era in how IT is taught in schools. Since then the Pi has been widely used in schools around the UK. But despite the success stories, teachers are still struggling to do anything with their Pi.

I normally suggest that they start small, and build on each project. I like to start with setting up the Pi as a lesson in itself. This enables the class to learn more about the board, and to use it safely. I would then move on to showing the class some basic Python code and connect up an LED to the Pi. I've found this to be a very successful goal and introduces lots of scope for further projects. The decisions made after this dictate the direction that each teacher takes with their Pi. Some teachers are using them in other lessons, such as CDT (Craft, Design and Technology) where robotics and product design are the focus, and some teachers are using their Pi in Science and Geography to gather weather data and create custom weather reports for their school.

2: Why Python? Language "X" is better!

For KS3 children are expected to understand and use two languages. One of which must be a textual language and this is where I feel Python is a natural fit. Python is a remarkably easy to understand language which is why it is ideal for learners. It's easy to use syntax and extensive documentation make it accessible to all. When used with a Raspberry Pi it opens up a world of projects with electronics and robotics.

Once the learner is comfortable with Python, they can use their underlying knowledge of coding concepts and apply it to learning another language. But one thing is certainly clear, with Python you can learn so much, so quickly.

3: But surely the Internet is the future?

The Internet has grown massively since it was widely introduced to the public in the late 1990s, and by this I mean the use of AOL and Freeserve CD's which fuelled "cheap" access for the public. In the time that followed we have seen changes to the tools used to create web content. Originally it was pure HTML (Hyper Text Markup Language) and this crude but effective language was pivotal in the development of the Web. We then saw the introduction of CSS (Cascading Style Sheets) and JavaScript.

These technologies are still with us and have been improved considerably over the years. In 2014 children will be taught more about the Internet, and HTML, CSS and JavaScript are seen as important components for their learning.

One area of this I have been working with is Google's Coder platform [1]. Coder is a sandbox web development environment built for the Raspberry Pi. In theory Raspberry Pi running Coder can be easily used in class to create a portable computing lab for children to rapidly develop and test their own web applications.

4: Open Source . . . Locked Equipment

The majority of teachers inform me that their school IT equipment and software are locked down and controlled by an outside contractor, and to install new software on their machines takes time and money. In one conversation that I had with a teacher, they informed me that they had been quoted £800 to install Python on the school's network, to which I audibly gasped. For £800 you could equip a computer lab with ten Raspberry Pi, and purchase all the extra equipment necessary to attach the Pi to your existing monitors (Pi View being the best solution to this issue).

I suggested that they try and use Portable Apps [2] to use many great free applications in the school, to which they replied that the USB ports of most school machines are disabled by their contractors, and that the loading of executable files is restricted to teachers only. While I see the logic behind this approach, I find it a poor solution to the issue and surely there are ways to introduce a level of granularity that permits the use of USB and executable files in a sand-boxed environment?

As you can see, there are many barriers facing teachers across the UK. Some are behavioural and these can be remedied by introducing the new concepts in a transitional manner, normally by comparing the previous method and illustrating the benefits of the new. Other issues are based upon the facilities and equipment that our school uses. Because of the "support" given by outside contractors, and they are normally very pro Apple or Microsoft, schools have indirect control of the applications and equipment used in lessons and this needs to be challenged. I believe the Raspberry Pi is an excellent user case to illustrate what can be done with Open Source technologies in our schools.

References

[1] `http://googlecreativelab.github.io/coder/`

[2] `http://portableapps.com/`

Building a Mendel90

Bob Clough

Part 2 – Electronics and Printing

Last time, we went through building the hardware side of the printer. Currently we have a lot of wires from motors, hotends, fans etc that we've been running through cable runs and zip tying out of the way. They all pop out in the electronics chamber, which is situated at the back right of the machine.

The first part of setting up the electronics is probably also the most convoluted, as it involves modifying a PC power supply to remove all the thousands of extraneous plugs and wires needed for powering a PC, capping off some, and joining together the others we need.

We started by snipping off all the connectors, leaving us with a nest of multicoloured wires. Next we fiddled with the wires a bit, to try and get them sorted by colour, to make it easier to pare them down.

The first thing we had to do was connect some wires together. We connected a red (5V) and black (GND) wire with a large resistor, which helps keep the power supply loaded, and keep the voltage correct. Next we connected the Green wire and a black wire, which tells the power supply to turn straight on when mains power is applied, and an orange wire to the brown wire, which allows the power supply's internal controller to know that it is outputting 3V correctly. These joints were connected to each other via another resistor, which keeps the 3V loaded similar to the 5V.

After this is done, we needed a high current 12 volt and ground feed for the Melzi control board. We twisted together all the yellow wires and all the black wires into bundles, and soldered together the ends. As the Mendel90 can pull 18 amps with all the heaters and motors on, its important to have as much current carrying capacity as possible, so we used all the wires.

After this, all that remained was to cut off all the unused wires and cover the ends of the cuts with heatshrink to stop them from shorting together.

Once the power supply conversion is complete, we moved on to mounting the electronics to the machine. The power supply itself is fixed to the bottom of the machine, and above it the Melzi board is mounted.

The actual connection was largely simple, first we connected the power feed, followed by the heated bed, endstops and the Y motor. The wires for the two Z motors are connected together before being connected to the board, so that they will both turn at the exact same rate and time.

The last part of the electronics assembly gets a bit fiddly. We basically have to split apart the ribbon cable into separate functions, and screw the tiny wires into the correct holes on the Melzi. I was finding this difficult, because I couldn't both see what I was doing, and get my hand in the machine to hold the wire in place and screw down the terminal, until I realized the shiny black surface of the Mendel90 could be used as a mirror to get a different angle on the connectors. After this the connections proceeded successfully.

At this point, the build is complete! We connected everything up, turned it on, and downloaded pronterface [1], which is a piece of free software written in python for controlling 3D printers. We clicked connect, and immediately had control over the printer. We were able to home and move all the axes, heat and cool the bed and extruder, and activate the fan.

The first step is to level the X axis. We did this by moving the carriage all the way to the left, placing a piece of paper under the axis, and moving the Z axis down in 0.1mm increments until the paper just gets trapped by the hotend. We then move the y axis all the way to the right, and turn the right hand leadscrew by hand until the paper just gets trapped again. At this point the X axis is pretty flat, at least close enough to try our first print!

First we cleaned off the glass bed with acetone. This helps remove any fingerprint smears, so our first layer will stick to the bed correctly. We selected the SD print option in pronterface, and chose android.g, which is a precompiled gcode file that comes with on the SD card. The printer whirred to life, and a small android logo was printed! [2] It popped off the bed a little bit on the antennae, which we fixed for the second print by applying a mixture of diluted PVA glue to the bed and letting it dry.

References

[1] **`https://github.com/kliment/Printrun`**

[2] **`http://hacman.org.uk/?attachment_id=322`**

Designing for Behaviour Change

Stephen Wendel
O'Reilly Media
ISBN: 978-1-4493-6762-6
400pp.
£ 30.99
Published: November 2013

reviewed by Lindsay Marshall

The subtitle of this book is "applying psychology and behavioural economics" which, if I were being cynical, could easily mean "selling stuff by manipulating people", but of course it's more complicated than that. There are loads of areas, particularly health, where it is beneficial to move people in certain directions without them necessarily realising that it is happening. So let's assume that none of the people reading this book are wanting to sell me flavoured, sweetened water or anything like that, because it seems pretty clear that the techniques described do work even if only in small ways.

The book itself is a little different from the usual O'Reilly books that pass by me – it's got a lot of colour illustrations for a start, and it is much more text heavy than usual, probably because it isn't filled with examples of code which do tend to make books look a bit more empty. And there is a lot in here, the book looks at all stages of designing a system that is meant to change behaviour, from basic principles about models of how the mind works with respect to change, through developing and implementing designs.

I must confess that I haven't finished reading the book yet – there is a lot in here, and it's interesting if jargony in places. If you work in UX design then you probably ought to read this, there are many good suggestions for how to go about designing solutions.

Network Security Through Data Analysis

Michael Collins
O'Reilly Media
ISBN: 978-1-4493-5790-0
348pp.
£ 30.99
Published: February 2014

reviewed by Paul Thompson

The biggest challenge with log data is that interesting events are rare compared with the overheads and costs involved in collecting, storing and managing the huge amount of data that *might* be needed to identify and investigate them. This book teaches techniques for deciding what makes sense to actually capture as part of an analytic trade-off.

Chapter 1 summarises the general principles of sensors and data collection, and defines the three domains of network, host and service. It explains how sensor placement (vantage) can affect data collection, such as which packets can be observed and how it can be affected by the network's routing infrastructure. It also discusses the impact that network layering has on instrumentation.

Chapter 2 examines vantage in more detail; how domains and actions might be deployed in real systems, and how it relates to the layers of the OSI and TCP/IP models. Consideration is given to the relative value of events from different sources: a high level event recorded in log data, for example, is preferable to the same event in network traffic which would have to be extracted from millions of packets – which can often be redundant, encrypted, unreadable or have been manipulated. On the other hand, network-captured events are the best method of identifying any auditing blind spots. Either way, judicious filtering, rolling buffers and snap length reduction are practical requirements.

This chapter includes some example `tcpdump` recipes, and leads neatly into a thorough overview of BPF filtering and macros. Given that filtering always imposes performance overheads, it was good to see alternatives discussed including traffic summarisation using Cisco's NetFlow and YAF (Yet Another Flowmeter) – which aggregates pcap packets into a flow (i.e. an approximation of a TCP session). A decent explanation was made of the suitability of flow and template-based NetFlow (IPFIX) - which includes useful fields for security analysis.

Chapter 3 considers sensors operating at the host or service domain and weighs up problems with logs, such as varying formats, locales and delimeters – which may necessitate an aggregator – and concerns about logs being a target for attackers.

Attention is given to methods of configuring common log formats such as for SMTP, and Apache's `mod_log_config` module, plus a discussion on logging levels and syslog transport protocols *facilities*. For HTML messages, the common log CLF (a single-line format) is examined with its *combined log format* variant (which includes the `Referer` (sic) and `User Agent` fields) – and compared with the extended (and expandable) ELF, which consists of a series of directives defining sequence entries. It is discussed how successful HTML logging involves deciding which of the >100 headers are actually worth tracking (probably `Cookie`, `Host`, `Referer`, `User Agent`).

This chapter also highlights the difficulties of relating a discovered event to the same event captured in other domains, but despite such challenges it does reinforce the view that an effective security system must combine network data for a broad scope, with logs for fine detail. For me, particularly useful were the few pages dedicated to the characteristics of an ideal log message, with suggestions for building descriptive messages (like journalists' *who, what, when, where, why, how* questions) for ensuring events can be related to information from other sources. Identifying the same phenomenon from different sensors (which is almost always involved when investigating security events) requires special considerations such as time being in sync, which are covered here too.

Chapter 4 addresses the mechanics of storing data and the basic problem in security analysis of how to quickly process large volumes of scattered information. As patterns emerge during the analysis of a problem, choices can be made about the additional data required to be extracted from the repository for further analysis, but such repeated queries and accesses can be expensive in time and resources. This chapter describes the fundamentals of engineering an efficient data system.

Approaches employed are: parsing flat files (simple, but lacking tools for optimised access); traditional databases such as Oracle and Postgres), and the emergent NoSQL paradigm involving a *big data* system using distributed platforms (like Hadoop), databases like (MongoDB and Monet) and specialised tools such as Redis and Apache SDLR. It is demonstrated how flat-file query systems can handle large netflows if well-designed (such as that developed by CERT for SiLK), e.g. with an indexed file hierarchy (partitioned by date, protocol, etc) to limit records read – necessary because of the I/O bound nature of log analysis. Whether to compress the files is also considered here.

RDBMS, employed in most persistent storage systems, suits a CRUD (create, read, update, delete) framework in which data integrity is important, but which comes at a performance cost. They are not suited for log data collection systems, where only sensors write to disk and only users read from disks – and where metadata management and integrity mechanisms are unnecessary.

This chapter briefly introduces NoSQL systems, but we are warned not to underestimate its downsides, such as the need for significant investments in hardware, system administration and heavy-duty programming. It describes the concepts of MapReduce: mapping – the independent application of a function to all elements in a list and reducing – combining a list's consecutive elements into a single element. Map operations are implicitly parallel because the mapped function is applied to each list element individually - so fits with big data / distributed data storage architecture, that relies on massive parallelisation. Three major categories of storage are defined: Key stores, effectively a giant hash table in that an entire data structure is associated with a key (employed by MongoDB, Cassandra); Columnar databases, which split each records across multiple column files with same index (MonetDB, Sensage), and Relational databases, where RDBMS stores complete records as individually distinguishable rows (MySQL, Postgres, Oracle). The most appropriate storage architecture depends mainly on the type of data collected and the type of reporting against it, and this chapter includes clear advice for making that selection.

The next five chapters discuss specific tools used for data analysis, with particular focus on SiLK and R.

Chapter 5 explains how the SiLK (System for Internet-Level Knowledge) suite of tools (the CERT NetSA Security Suite at **`http://tools.netsa.cert.org/silk`**), each performing a specific query, manipulation or aggregation of data, can be piped together to quickly and efficiently query very large volumes of network traffic. Examples are provided demonstrating how addresses/protocols can be filtered from flow data (with `rwfilter`) and how records can be manipulated and formatted (`rwcut`) plus basic examples for the other main tools in the suite (`rwcount`, `rwuniq`, `rwset`, `rwbag`, etc).

The standard flow collection software for the SiLK toolkit is YAF, and examples are given to read pcap data and live packets - demonstrating its sophisticated command-line options.

R (aka GNU S), the open source statistical analysis package, is the focus of Chapter 6 – and this should serve as a useful primer for anyone new to R – and a good basic tutorial on its language including functions, conditionals and iteration. For large datasets, R's strengths over clunky graphical tools such as Excel are clearly explained, like the ability to script table manipulation.

Data frames (ad hoc data tables, similar to arrays or hashtables in other languages) in which a column represents a single variable are probably the most useful structure for analysing network data. Detailed examples show the powerful facilities that R has for selecting, filtering and manipulating data frames, such as `read.table` for importing files in SiLK's `rwcut` format.

Some of R's visualisation capabilities are demonstrated by using `plot` to create time-series, histograms and bar charts including how to export them in usable formats.

Much of the process of statistical testing involves constructing alarms by identifying significant features – which requires specifying important attributes and behaviour. The last few pages of this chapter are dedicated to achieving that using R, which actually means testing null hypothesis claims – such as whether a particular dataset matches a distribution. Brief examples are provided including running Shapiro-Wilk and Kolmogorov-Smirnov goodness-of-fit tests to determine whether a sample is normally distributed; further detail is provided later in Chapter 10.

Event-based sensors are covered in Chapter 7. Analytically, passive sensors such as IDS and AV systems behave similarly to active systems like firewalls – in that they create *events* in response to data. The ability of an IDS to report only on specific phenomena inferred from its observed data is what differentiates it from a simple reporting sensor such as NetFlow. Implementing effective IDS thresholds and abusive activity triggers is difficult; the problem is not detection but context and attribution. The first part of this short chapter explains how IDS failings impact on analysis. Problems with IDS binary classification (i.e whether the data is normal or has characteristics of an attack) are summarised as *moral* where attacks are indistinguishable from innocuous or permitted activity; *statistical*, like IDS producing too many false positives, or *behavioural*, where attackers evade detection by keeping their activity to a minimum. Popular IDS systems are summarised, but the tone is pessimistic, particularly for those that don't interoperate. Security event management (SEM) software, effectively a database collating from multiple detection systems, is also discussed with examples including ArcSight, LogStash and Splunk.

The second part of this chapter discusses strategies for improving detection systems. Suggested improvements include combining with whitelists; improved detection rules (snort's rule language is given good coverage), and pre-fetching summary data to provide contextual visualisations to the analyst.

An alert or log file event will usually require *walking up* the OSI stack glean more information about the source, and Chapter 8 covers the tools for doing that. Methods discussed include obtaining geographic location from reverse DNS lookups, IP intelligence services (like MaxMind's GeoIP, which has useful APIs) and Neustar; IP address ownerships from IANA reservation hierarchy; manufacturer OUI codes from MAC addresses, etc. Much of the chapter is taken up with explaining IP addressing structure and the use of common tools like `dig` and `traceroute`. DNS blackhole list (DNSBL) services including SORBS, Spamhaus and SpamCop are also described.

Chapter 9 is really a round-up of some tools useful for visualisation (Graphviz), probing (`netcat`, `nmap`, Scapy), and packet inspection (Wireshark). Examples of their use are quite comprehensive, but I doubt they are required for the book's target readership.

The remainder of the book is focused on the actual analysis of collected data. Chapter 10, about exploratory data analysis (EDA), is essential reading. It promotes the approach of examining a dataset without any pre-conceived assumptions about the contents.

Basic goals of data analysis for information security are alarm construction (generating a number and comparing it against a model of normal activity); forensics (determining more information about a given datum, such as an address); defence construction (constraints on attackers without burdening users), and situational awareness (knowledge management such as inventory and past history data). Those goals are discussed around the need for a structured workflow. The first EDA technique is typically to visualise the data to identify significant features. Fundamental categories of statistical variables (interval, ratio, ordinal, nominal) are reviewed and different visualisation techniques – that should be selected based on the type of variable being measured – are demonstrated with examples: univariate visualisation using R to plot histograms, bar, quantile-quantile plots, etc – and multivariate visualisation, which is more of a technique than a specific set of plots. Advice is provided on clearly presenting information from each of several datasets.

The next phase of the analytic process, *operationalising security visualisation*, requires modifying visualisations to expose events. Recommendations include disabling auto-scaling on plots (that can mask anomalies); labeling anomalies, and using trend lines to distinguish artifacts from observations.

Chapters 11 to 14 examine the techniques and behaviour of attackers, and how such knowledge can be used to construct alarms and be used for forensics and investigation. Chapter 11 about the methods by which attackers *fumble* around a network – and how such behaviour can be differentiated from that of legitimate users. Attack models are used to describe steps undertaken in a majority of attacks: *reconnaissance* is when a target is scouted, e.g. by social engineering or `nmap` scanning; *subversion* is the launching of an exploit and taking control; *configuration* is the conversion of a system such as disabling AV or installing malware; *exploitation* is the use of the host for the original reason for targeting it, and *propagation* is the use of the compromised host to attack other hosts. Tell-tale network events indicating each of those attack models are examined, such as the failed TCP connections (TCP state machine) left by port scans, and advice is given on raising alarms.

Chapter 12 examines the phenomena that can be identified by comparing traffic volume against the passage of time. Such phenomena include beaconing (regular contacts with a host, as is seen by bots polling a command server), file extraction and DoS – but the noisy nature of regular traffic means there's rarely a significant relationship between the number of bytes and the importance of the events. Techniques presented include plotting an organisation's normal workday schedules to highlight anomalies; running a *beacon detector* script (made available on Github) - which checks median distance between equally-sized dumps of sequences of flow records, and running a *raid detection* script that compares traffic originating from a host with thresholds based on normal operation over time. Viable techniques for handling various forms of DoS attacks are also suggested.

Chapter 13 explains how particular components of network traffic logs are suitable candidates for conversion to graphs. Advice given for converting raw data into graphs, and examples for consideration include using IP addresses as graph nodes (*vertices*) and the existence of a flow between them used as links (*edges*), or the individual pages in HTTP server logs being treated as nodes linked together by `Referer` headers. Scripts are provided to simplify the task of creating graphs from lists of pairs (i.e. `rwcut` output).

Analysing graphs is also covered in this chapter. An example is using component analysis as an alarm for identifying an attacker that artificially links discrete network components, such as by attempting to establish SSH sessions with hosts that are unrelated (e.g. in separate parts of the organisational structure – abnormal behaviour for a typical user).

The challenge of identifying applications in logs and network data is covered in Chapter 14. No longer is it a matter of simply checking port numbers or header packets, particularly as modern application developers and users may have even prosaic reasons for hiding traffic classes (BitTorrent) or encrypting payloads (privacy). Mechanisms for application identification discussed in this chapter include OS fingerprinting and banner grabbing (with scripted examples using `netcat` and `scapy` as an alternative to `tcpdump`). Application identification by behaviour, useful in the absence of payload, is achieved by examining an application's features such as packet size and connection failures (again, useful example scripts are provided). Application identification by subsidiary site involves examining the *magnet links* polled – for example BitTorrent users may visit well-known torrent sites (Pirate Bay, etc) in order to determine the actual filenames that are subsequently downloaded from decentralised peers.

The final chapter, 15, presents strategies for handling false positives. The importance of situational awareness (particularly the inventory process) is emphasised for achieving an efficient anomaly-detection strategy rather than simply reacting to matched signatures, etc. It is suggested that an initial network inventory map be produced by a combination of technical techniques and tapping network managers' local knowledge. The initial phase should be to identify the space of IP addresses being monitored, then each progressive phase should partition that space into various categories. Importance is placed on the need to identify dark space and unmonitored routes (i.e. legitimate routes where traffic is not being recorded) - and suggestions are made for doing that using `rwfilter`, `rwuniq`, etc. Other network mapping challenges covered include NATs, VPN traffic and dynamic addresses. Tips are given for coping with the ever-changing nature of a network infrastructure, and how to implement a continuous audit approach towards keeping the inventory updated.

The book mostly focuses on I/O bound data, i.e. analysis entailing identifying the correct data to be read then extracting it – rather than the traditional approach of log-file parsing, which is relatively time-consuming on a large scale and where a query could take longer to execute than it took to collect. Attention is given to the importance of a well-designed storage/query system with an emphasis on efficiency of disk space and query times. This often means trade-offs have to be made, for example in the case of deciding whether completeness of collected data is more important than redundancy of data.

I was surprised to see little mention of transport and data protocols besides IP over Ethernet. Particularly of interest to many would have been Fibre Channel, given its widespread use as the backbone for SAN implementations. More discussion of streaming processing would have been useful too; although not appropriate for exploratory analysis, it does fit in network security analysis for situations where real-time processing of pre-defined alarms is required.

Over all I found there was the right balance of dialogue in the book compared with technical instruction, which made it an enjoyable read rather than just a how-to manual. Sufficient guidance is provided to help the reader decide which tools and techniques are most appropriate for evaluating based on their specific environment, and very good tutorial-style overviews are given for getting started using them. Each chapter includes recommendations for further reading, and I found the sample code snippets provided on Github to be useful. I didn't like the use of active voice and personal pronouns (quite distracting), but I didn't find many errors (none serious) and there were not enough Americanisms to be annoying. I'll certainly be recommending this book to my colleagues.

Android Developer Tools Essentials

Mike Wolfson
O'Reilly Media
ISBN: 978-1-4493-2821-4
250pp.
£ 18.99
Published: August 2013

reviewed by Gavin Inglis

Books with impressive titles have always been useful for more than simply reading. If a gullible friend sees "Android Developer Tools" and believes you are building a robot, see if you can convince them that it will eventually resemble the glass-eyed cassowary on the cover.

This book is of course nothing to do with robots. It concerns Android, the popular mobile platform, and specifically its developer tools, which are managed separately from the core OS. The book is a relatively slim volume at 232 pages. Don't expect to learn programming from the text; it assumes the reader is an experienced Java coder who wants to get up to speed on the tools available for Android development.

We begin with the Android stack: Java JDK, Android SDK, Eclipse IDE and ADT plug-in for Eclipse. The text gives lip service to developing without an IDE or using an alternative to Eclipse. There are installation instructions, but these are sensibly brief, and focus on making sure that you have all the components in place rather than a step-by-step "now click here" approach.

Things become more interesting with Chapter 3, on configuring devices and emulators. Here we learn how to create multiple emulator images for different Android devices, and the options and compromises required. Also explored is how to connect an actual piece of Android hardware and the various benefits that this offers. Getting hands-on with the emulator, there are some useful tips like keyboard shortcuts, or using snapshots to speed up a slow launch.

Two chapters follow on the alternatives for an integrated development environment. Eclipse is first, with specifics on using the layout, templates and wizards. At sixteen heavily illustrated pages this is brief, but it does include good links, such as how to create your own templates.

Android Studio, a likely direction for Android development, was announced in May 2013, when this book was surely deep into the production process. The relevant chapter was written by guest author Donn Felker. Although perfectly clear, and a natural fit with the rest of the book, it was written relative to version 0.0.5 (Android Studio is up to 0.4.4 at the time of review.)

The middle of the book comprises good nuts-and-bolts stuff, and covers logging, debugging and finding potential issues in the code. Simulating mobile use gets a welcome look: you may travel via GPX file, fake some telephony or record sensor activity to repeat later.

By this time you will be thinking about releasing your new app to the world. Chapter 9 concerns build tools and the packaging/signing process. It covers building with Ant, and obfuscating using ProGuard. Also included is basic use of Gradle, and a passing reference to Maven.

The final three chapters consider the user interface: using the XML editor to define layouts, using graphical assets, scaling them to create icon sets and previewing on different sized screens. The book closes with some advice on optimising the layout hierarchy and taming the delightful Application Exerciser Monkey.

“Android Developer Tools Essentials” is nicely pitched towards a Java programmer with some experience. It knows its audience well. The text includes the important material and keeps the book compact by skipping over what should be obvious. This means it might lack detail for some potential readers, but it does get the essentials into a clear and usable form. Much of its value is concentrated in the shortcuts, gotchas and links scattered throughout the text.

The conclusion promises updates at the author’s website, but this reviewer could only locate photographs of his wedding, dog and Pez collection.

The pace of Android development means the text will date quickly, but it’s not intended as a long-term reference. It’s a guide to the options available and a stepping stone to help you become familiar with the developer tools. There are a generous number of screen captures, showing menus and layouts to keep everything clear. The flipside of this is that the book may not offer much if you are already developing Android applications.

Contributors

Bob Clough is a Founder Member of Hackspace Manchester (`hacman.org.uk`) and Manchester’s 3D Printer User Group (`3dpug.co.uk`).

Gavin Inglis works in Technical Infrastructure at the EDINA National Data Centre in Edinburgh. He is a teacher, photographer and musician who has recently discovered the joys of spreadsheets as a recreational tool.

Kimball Johnson is a Developer at Shadowcat Systems in Lancaster. He has been programming since a very early age, starting with BBC Micros, then MS DOS and Windows Systems, however was enlightened with a copy of Debian GNU/Linux Woody at university. From this developed an affinity to Systems Administration, but like all good admins he is lazy and so tries to automate as much of his job as possible, and now follows the way of the Infrastructure as Code.

Mark Keating is the Managing Director of Shadowcat Systems and a keen contributor to an array of community projects. He describes himself as an adminion organiser, cat herder, writer and photographer. He lives in Lancaster with two children, a cat and tropical fish. Mark has heard of the concept of ‘free time’ but doesn’t like to indulge in it. Mark can be found on Twitter as `@shadowcat_mdk`. His company blog is at `http://shadow.cat/blog/mark-keating/` and other links can be found at `http://mdk.me/`.

Holger Kraus is a trained electronics engineer who fell in love with the Internet in the mid nineties. Since then he has gained extensive experience as technical architect/sysadmin with online insurance and travel technology, and now works as a web systems manager specialising in secure and reliable operations of domain name, mail, database, and web services based on open source software.

Lindsay Marshall developed the Newcastle Connection distributed UNIX software and created the first Internet cemetery. He is a Senior Lecturer in the School of Computing Science at the Newcastle University. He also runs the RISKS digest website and the Bifurcated Rivets weblog.

Jane Morrison is Company Secretary and Administrator for UKUUG, and manages the UKUUG office at the Manor House in Buntingford. She has been involved with UKUUG administration since 1987. In addition to UKUUG, Jane is Company Secretary for a trade association (Fibre-optic Industry Association) that she also runs from the Manor House office.

Les Pounder works closely with North Western Linux and Free Software groups to promote the use of Open Source software as opposed to proprietary software. He is also the organiser of UCubed, a free Linux and open source event in Manchester, and an organiser of Barcamp Blackpool and Blackpool Geekup, and has been head of crew at Oggcamp. He writes for Linux Format magazine and contributes to Linux podcasts including Fullcircle, Ubuntu UK Podcast and Linux Outlaws.

Paul Thompson is based in London and works for SUSE as a Linux technical consultant specialising in the financial sector. He is also an aspiring sourdough baker and heliciculturist, and can be reached on Twitter at `@raganello`.

Roger Whittaker works for SUSE supporting SUSE Linux Enterprise Server for major customers in the UK. He is also the UKUUG Newsletter Editor, and co-author of three successive versions of a SUSE book published by Wiley.

Contacts

Kimball Johnson
UKUUG Chairman
Preston

Gavin Atkinson
Council member
York

Tim Fletcher
Council member
Manchester

Holger Kraus
Council member
Leicester

Ian Norton
Council member
Rochdale

Stephen Quinney
Council member
Edinburgh

Quentin Wright
UKUUG Treasurer
Warwick

Roger Whittaker
Newsletter Editor
London

Alain Williams
UKUUG System Administrator
Watford

Sam Smith
Events and Website
Cambridge

Jane Morrison
UKUUG Secretariat
PO Box 37
Buntingford
Herts
SG9 9UQ
Tel: 01763 273475
Fax: 01763 273255
office@ukuug.org