Module Final report

Marcus Sung

CS7NS1/CS4400 Module Final Report

- Student name: Marcus Sung
- Student number: 14335274
- Date:

InfernoBall team self-evaluation

- Team number: 11
- Team members:
  - Marcus Sung
  - John Aitling
  - Xuming Xiu
  - Sahir Sharma
- Insert team member names/initials in column headers of table below
- Effort must be high/medium/low/zero
- Effectiveness must be high/medium/low/zero

Use zero if someone really didn't participate at all. Include your evaluation of yourself as well as other team members. Be honest.

	Marcus	John	Xuming	Sahir
effort	high	medium	low	low
effectiveness	high	low	low	low

What I learned (1-2 pages)

I learned various things from going to lectures (normal and guest) and doing the written assignments, and the practicals.

Lectures

- What I learn about Scalable Computing (general):
  - Things to look out for in scalable computing: Scalability, adaptability, dispersibility, accessibility, affordability, and reliability.
  - This helped when reading the papers for the written assignments.
- What I learnt about Internet of Things:
  - IoT and its structure.
  - IoT nodes and their constraints (low resources, low computing power, low memory etc)
  - Architectures of IoT systems.
  - Applications and the benefits and challenges.
  - Industry 4.0
- What I learnt about processing units:
  - The differences between CPUs and GPUs.
  - The cases in which each one is good for.
  - Advantages and disadvantages of them.
  - CPU and GPU orientated programming (OpenCL).
  - OpenCL specific programming approach.
- What I learnt about passwords:
  - Advantages and disadvantages of passwords
  - Best practices in managing and creating passwords
  - Password alternatives e.g biometrics
  - Cryotographic hashes, how they work and the different kinds
  - Some ways to attack a hash and crack it
- What I learnt about failures:
- Different causes of failure
- Inevitable but can learn a lot from them
- What happens when a failure occurs and how they are dealt with
- How to prevent them

Assignments

This is what I learnt doing the written assignments:

- Assignment 1 (Scalable Computing):
  - Application of IoT in personal healthcare:
    - RFID technology
    - Environmentally passive sensors and body centric sensors
    - Privacy, safety and reliability concerns
  - Smart Cities:
    - Architecture
    - Features (possible)
    - Technology involved in realising the concept (Already done in Padova)
- Assignment 2 (CPU and GPU):
  - Language based approach
    - Parallelism and reliability integrated into a language (MISO)
    - Benefits of language based approach for low powered computing
  - Energy efficiency in clustered many-cores:
    - Programmable tightly-coupled clusters of processors.
    - Pros and cons of using it.
    - Applications.

- Assignment 3 (Cloud/Edge or Fog Computing):
  - Mobile edge computing:
    - How MEC works and the architecture.
    - Pros and cons of using MEC.
    - Limitations on research in particular security.
  - Fog Computing and IoT applications:
    - Security and privacy not researched enough.
    - Advantages of using fog computing.
    - Limitations of fog computing.
    - Architecture of fog computing.

Practicals

Doing the practicals I learnt the following things.

- Practical 1(Compiling John The Ripper):
  - How to register with rosettahub
  - How to create and stop an instance and how to SSH into it
  - How to clone into github and compile tools such as John The Ripper
  - How to run bash scripts (I had no previous knowledge on scripting of any sort)
- Practical 2(1k passwords):
  - Different hash types and formats
  - The various algortithms and methods that JTR could use
  - Methods such as brute-forcing, dictionary attacks, mask attacks, rules etc
  - Salted vs unsalted hashes differences more clearly
  - Sending files to the instance and retrieving files from the instance (scp)
- Practical 3(More hashes):
  - How to set up a GPU instance
  - How to save images on rosettahub
  - Compile and run hashcat
  - Various different hashcat modes
  - Difference between GPU and CPU cracking
  - Implementing rules attacks, combinator attacks, prince attacks and mask attacks on hashcat
  - Writing scripts in python and bash to automate tasks
  - Using various command line tools such as sed, grep, awk etc to edit text files in turn automating tasks
  - Using tmux to leave instances running in the background
  - Slow and fast hashes differences
  - Multiple GPU vs singl GPU differences
  - Ram management when dealing with different methods so you don't run out of memory
  - Optimising attack methods e.g using word list compiled of all mask attack iterations is a bit faster then using mask attack purely
  - Vertical scaling and horizontal scaling
  - Using Google cloud scaled vertically (e.g adding more power to a single instance)
  - Running AWS and Google Cloud at the same time offered horizontal scaling (running more machines)
  - Collobaration can help speed things up
  - Slow hashes such as argon were very hard to crack in a reasonable time since they could only be cracked using CPU attacks
- Practical 4 (Group assignment):
  - How Shamir sharing worked
  - How to be more efficient working in groups cracking hashes
  - How to work with a team that did not put much effort in or were not that knowledgable in the module
  - Sometimes it is quicker to do things yourself instead of waiting for others to do it (we dropped from the top 5 in the leaderboard to the bottom 5 whilst I waited for them to do some tasks)
  - Splitting hashes between many would be the ideal way to crack faster

What I did (2-4 pages)

Lectures, written assignments and class test

I went to the lectures (normal and guest) and took notes in each one. I would use these notes to look out for key concepts needed for the written assignments when reading the papers provided. Using these notes and written assignments I made a cheat sheet for the class test. I also revised the papers I read and some of the others. Some of us created a google drive where we shared all of our written assignments so we could revise for the test using them for revision. I also attended the guest lecture on developing a OpenCl program and participated in it.

Practicals

The following sections contain the steps I undertook to complete the practical assignments. There is a scripts section linking the scripts I wrote and what they do.

Practical 1

- Registered for rosettahub and followed the e-mail instructions to receive credits
- Followed the instructions on how to create an AWS instance and got one running
- Downloaded the keys needed to SSH into the instance
- Cloned JTR jumbo and followed install instructions
- Sent the benchmark script to the instance via scp
- Ran the script and downloaded the output via scp onto my local machine
- Submitted it to submitty

Practical 2

- Looked up various methods of John The Ripper
- Set it up locally to test rather before attempting to use it on the instance
- Downloaded the hashes file from the e-mail and examined the hashes
- Tried to see if there was a way to distinguish different hashes from one another and saw that there was a pattern
- Used a hash identification (hashtag) tool to get a better idea of the hashes
- Spent a lot of time researching he different methods to try and see what was best before testing anything
- Decided to try RockYou word list as the first attempt with Jumbo Rules set from JTR and cracked all the passwords locally quite quickly
- Realised Jumbo Rules had slowed it down and would have cracked it faster using without rules

Practical 3

- Set up a GPU instance (followed instructions github) and got hashcat compiled
- Created an image of this instance so as not to have to replicate the steps again
- Set up hashcat locally first to test various methods before pushing to the instance for more speed
- Spent some time researching the attack types and modes of hashcat
- Used JTR, and hashtag allowed for easier identification of the hash types
- Discovered the various hash types (des, md5, sha256, sha512 and argon) and split the hash file by the hash type
- Found that speeds ranged from des being the fastest and argon the slowest
- Knowing this des was used to test for the different types of passwords
- Rockyou was tested on des and cracked the first few fairly quickly
- While trying to find out the other password types rockyou dictionary attack was performed on the next fastest and so on to make sure there was always something being cracked and reduce overall time
- Bruteforcing and testing various different dictionaries revealed that 5 letter words appeared quite often
- Bruteforcing 5 letter words was then tried and a further pattern found that it was lowercase only
- Knowing this a mask attack was able to be performed

- After cracking more passwords of the 5 letter lowercase another pattern was revealed to show that there would be at least one vowel contained in the words
- Mask attacks were effective until the slower hashes
- It was then found that generating a word list from the masks resulted in being able to perform a dictionary attack which was quicker
- Again there was always one type of password being cracked whilst trying to figure out the other type
- When looking at the bruteforced des passwords again another pattern was found this consisted of 2 four letter words combined e.g boat and fish -> boatfish
- Research was done into how to perform an attack that could take into consideration this pattern
- Combinator attacks were found to work on hashcat and this was done
- This attack mode took in two dictionaries and combined the words from both into one and tested it against the hashes
- The dictionary used was the unix words dictionary with everything but the 4 letter words filtered out
- This performed sufficiently for the faster hashes but became slower
- To combat this a dictionary again was formed instead of using the attacks which helped speed it up a good amount
- After significant hashes were cracked each time a dictionary of cracked passwords was made and run through the hashes again to catch any ‘low hanging fruit’
- PBKDF and ARGON were the slowest to crack
- PBKDF in the form we received was not supported on hashcat so it had to be altered using a script to a form that hashcat recognised
- However it was found that hashcat actually missed some hashes doing this
- Instead PBKDF was done on JTR using a GPU supported mode that was found
- Argon is not crackable using a GPU type attack so it all had to be done on JTR
- This was found to be extremely slow for all password types
- Splitting the dictionaries used into smaller pieces and randomly trying these pieces offered some passwords up
- The most passwords cracked was using previously cracked passwords
- Collaboration between a good portion of the class was done and we combined all of our cracked passwords into one dictionary
- This provided even more argon passwords
- All the cracking was done both locally and on instances
- It was done so there would be little downtime between cracking anything
- Using Tmux allowed for the instances to run in the background without worrying about failure causing it to stop
- There was a lot of trial and error in finding the best methods to crack the passwords
- One method attempted was to run JTR on more than 16 computers cracking argon by splitting up the wordlists between them
- This did not provide a significant increase in speed for any password type
- It was found that Google Cloud provided a significantly more powerful GPU instance so this was used for the majority of the slower hashes
- Various scripts were used to automate task and will be listed in the last section

Group project

- The assignment was read and the infernoball was downloaded from the e-mail
- The first task was to check how the layout of the infernoball looked
- For the most part it was JSON
- After the code that was used to create the infernoball was analysed
- A script was created to split the JSON file into the separate files (shares, hashes and cipher)
- It was found that there was various hash types again (sha1, sha512, pbkdf and argon)
- These were once again split into separate files by the hash type
- Rockyou was used to test the fastest hash and was found to be the password type for the first layer
- Again the next fastest was then tested and so on
- Whilst some of the passwords were cracking the code used to generate the infernoball was looked at again to try and change the code to decrypt the next layer
- The secret was generated using a script

- The decrypt script was made and the next layer could be decrypted if given the correct secret
- Upon decrypting the next layer it could be split into the same files but an extra one was found which gave a clue to the password type of that layer
- A group meeting was then held and the scripts were shared amongst the others
- We sat together to figure out the next layer and found it to be the compound words from the previous assignment
- We then decided the best way to crack faster was to split the hashes amongst one another and crack them and make sure to upload any passwords cracked to test them for the secret
- After this meeting there was no other group meeting
- I ended up figuring out the majority of the layer clues with help from one other team mate on 1 or 2 levels and cracked the majority of the passwords alone
- I did all the decryption of each layer and submitted them all
- At each layer the clues were took a while to figure out but after figuring them out it was pretty straightforward in cracking the passwords
- Google Cloud was used for the majority of the passwords due to the faster speeds with AWS being used for argon as there was instances with higher number of CPU cores
- The same methods that were used in the previous assignment were used here

Scripts

The scripts are divided between the individual practical scripts and the group project scripts.

Individual

- Link to repo containing all the scripts: https://github.com/sungmarcus/scalable-computing/tree/master/scripts
- Link to description of what each script does: https://github.com/sungmarcus/scalable-computing/tree/master/final_report/scripts_desc.md

Group

- Link to repo containing all the scripts: https://github.com/sungmarcus/scalable-computing/tree/master/inferno_ball/scripts
- Link to description of what each script does: https://github.com/sungmarcus/scalable-computing/tree/master/final_report/scripts_desc.md

Module evaluation (1 page)

Like

- The topics were interesting
- Guest lectures were good and a nice change of pace to normal lectures
- The practicals were fun as I learnt many new things (scripting, working with command line, hash cracking)
- The written assignments were good in terms of learning new topics as they made you read papers
- The written assignments format (only one page) I felt was perfect as it did made you get straight to the point of what you learnt
- The competition element of the practicals were a good motivator
-

Dislike

- The time spent on the practicals was too much
- The competition was good but perhaps bonus points for the first one to complete was not the right idea as this meant pushing aside all other module work and devoting a lot of time initially to get the early lead
- The lack of notes made it difficult to revise or catch up on material you may have missed
- Scalable computing could have been more clearly explained or approached at the start as it took a while for me to understand the scalability aspects in some of the material and practicals
- The AWS intances were good up to a certain point but then everyone found google cloud
- There should be more explanation on the best way to solve the practicals so people who may not have done so well could learn something and the people who did well could have also seen if their method was correct
- People took up all the computers in the labs cracking as they did not seem to know how to run the programs in the background on the instances or were just too cheap and didn't want to spend their credits
- The above meant I had limited access to computers I needed for a certain module (even though it was said that its okay to log them out it, there was still the case of people complaining to you if you did log them out, so people may not have been as inclined to log someone out)