https://github.com/nating/cs7ns1/blob/master/MODULE-FINAL-REPORT.md

CS7NS1/CS4400 Module Final Report

- Student name: Geoffrey Natin
- Student number: 14318196
- Date: 15/11/18

InfernoBall team self-evaluation

- Team number: 18
- Team members: - Abhishek Jain, James Farrell, John Lincoln, Geoffrey Natin

All team members made good contributions. After initial group contact through email, we communicated through WhatsApp. For the most part, all team members were contributing to the conversation every day from the day that our InfernoBall was delivered until the day that it was finished. Communication was round the clock, from very early in the morning to very late at night. My phone was constantly buzzing with updates on hashes cracked for almost two weeks.

Some members were better at finding and narrowing down our wordlists, some members were better at running cracking constantly with almost instant turnovers with attack methods. In my opinion, each member of the team put in an admirable effort and was effective with the work that they did. As group projects go, this one went down extremely smooth. I believe myself lucky to have been paired with the three students that I was, and I think that each of them deserves a good grade for the effort that they put into this project.

	AJ	JF	JL	GN
effort	high	high	high	high
effectiveness	high	high	high	high

What I learned

What I learned Directly

- How to use Hashcat and JohnTheRipper.
- How to use RosettaHub.
- The different kinds of instances available on AWS.
- The properties and challenges of Internet of Things computing.
- What OpenCL is and how to use it for writing scalable programs.
- The different types of processing units, their differences and the challenges associated with them.
- The different types of machine maintenance.
- The technology pillars of Industry 4.0.
- The properties of Fog and Edge computing.

What I learned Indirectly

- How to set up multiple accounts on Google Cloud Platform.
- How to request limit increases on Google Cloud Platform.
- How not to set up multiple accounts on Google Cloud Platform.
- How not to set up multiple Google Accounts.
- The different kinds of instances available on Google Cloud Platform.
- More unix commands.
- Different kinds of hashes.

Practical 5 - InfernoBall

In order to get this section down to 4 pages (as set in the report specification), only work done in practical 5 is described. Practically all work done in previous assignments was repeated for practical5 anyway. Work done in the other practicals is described at https://github.com/nating/cs7ns1/tree/master/practicals/practical3 and https://github.com/nating/cs7ns1/tree/master/practicals/practical4.

Please note that although the story is told here, it might be easier/better followed browsing the 'practicals/practicalX' files at https://github.com/nating/cs7ns1, which contains all scripts, wordlists, and other files discussed in the following pages.

Outset Steps taken

- Created a script for reading in an InfernoBall and a potfile and checking if the secret to crack the InfernoBall's outer layer could be found.
- Created a script to crack an InfernoBall's outer layer and generate a new layer with a new directory and infernoball.json & hash type files. This way, whenever more hashes were cracked, we could skip over most manual work involved in changing layers with a simple command.
- Set up Google Cloud instances with as many NVIDIA TESLA V100 GPUs as possible.

Layer 1

- Ran `hashcat -m 15100 layer-01/hashes/sha1crypt.txt rockyou.txt` to crack some hashes to start with (rockyou.txt is usually a good starting place to determine some pattern).
- Noted that every password was found in rockyou.txt.
- Ran a wordlist attack with rockyou.txt on PBKDF2 and SHA512 hashes as well.

Layer 2

- To to crack some hashes to start with (rockyou.txt is usually a good starting place to determine some pattern), ran:
  - `hashcat -m 15100 layer-01/hashes/sha1crypt.txt rockyou.txt`
  - `hashcat -m 1800 layer-01/hashes/SHA512.txt rockyou.txt`
  - `hashcat -m 10900 layer-01/hashes/PBKDF2.txt rockyou.txt`
- Noted that all passwords were two 4 letter words appended together, the second word was always uppercase and that no words contained non-alphabetical characters.
- Added combinator wordlist creation code to run.py.
- Created a combinator style wordlist of uppercase four letter words words from the linux

- Ran:
  - `hashcat -m 15100 layer-01/hashes/sha1crypt.txt 4-letter-combinator.txt`
  - `hashcat -m 1800 layer-01/hashes/SHA512.txt 4-letter-combinator.txt`
  - `hashcat -m 10900 layer-01/hashes/PBKDF2.txt 4-letter-combinator.txt`

Layer 3

- To to crack some hashes to start with (rockyou.txt is usually a good starting place to determine some pattern), ran:
  - `hashcat -m 15100 layer-01/hashes/sha1crypt.txt rockyou.txt`
  - `hashcat -m 1800 layer-01/hashes/SHA512.txt rockyou.txt`
  - `hashcat -m 10900 layer-01/hashes/PBKDF2.txt rockyou.txt`
- Noted that format was five character passwords with a digit occurring in one of the last three positions and alphabetical characters making up the rest of the password.
- Ran `?l?l?l?l?d`, `?l?l?l?d?l` and `?l?l?d?l?l` mask attacks for sha1crypt, SHA512 and PBKDF2 passwords.

Layer 4

- Took the hint, and used CeWL to run `cewl --write wordlists/cewl-scss.txt https://www.scss.tcd.ie/` and `cewl --write wordlists/cewl-tcd.txt https://www.tcd.ie/`.
- Ran:
  - `hashcat -m 15100 layer-01/hashes/sha1crypt.txt wordlists/cewl-scss.txt`
  - `hashcat -m 15100 layer-01/hashes/sha1crypt.txt wordlists/cewl-tcd.txt`
  - `hashcat -m 1800 layer-01/hashes/SHA512.txt wordlists/cewl-scss.txt`
  - `hashcat -m 1800 layer-01/hashes/SHA512.txt wordlists/cewl-tcd.txt`

Layer 5

- Remembered that at one stage, I had access to all submitty usernames due to a period where students could 'pick' their teammates but not really.
- Found that all usernames are presented in a script tag on the team page.
- Extracted student usernames from JSON array on submitty.
- Ran:
  - `hashcat -m 15100 layer-01/hashes/sha1crypt.txt wordlists/usernames`
  - `hashcat -m 1800 layer-01/hashes/SHA512.txt wordlists/usernames`
  - `hashcat -m 10900 layer-01/hashes/PBKDF2.txt wordlists/usernames`

Layer 6

- To to crack some hashes to start with (plain rockyou.txt is usually a good starting place to

determine some pattern), ran:

  - `hashcat -m 15100 layer-01/hashes/sha1crypt.txt rockyou.txt`
  - `hashcat -m 1800 layer-01/hashes/SHA512.txt rockyou.txt`
  - `hashcat -m 10900 layer-01/hashes/PBKDF2.txt rockyou.txt`
- Noted that all passwords seemed to be between 6 to 8 characters long.
- Cut rockyou.txt down to only passwords that are between 6 and 8 characters long:
  - Ran `awk 'length($0)>5' rockyou.txt > temp.txt` to get rid of passwords less than 6 characters long.
  - Ran `awk 'length($0)<9' temp.txt > wordlists/rockyou-subset.txt` to get rid of passwords more than 8 characters long.
- Ran:
  - `hashcat -m 15100 layer-01/hashes/sha1crypt.txt rockyou-subset.txt`
  - `hashcat -m 1800 layer-01/hashes/SHA512.txt rockyou-subset.txt`

Layer 7

- Searched the internet for "keyboard patterns wordlist" and downloaded a wordlist from danielmiessler/SecLists/Passwords/Keyboard-Combinations.txt.
- Ran:
  - `hashcat -m 15100 layer-07/hashes/sha1crypt.txt wordlists/keyboard-patterns.txt`
  - `hashcat -m 1800 layer-07/hashes/SHA512.txt wordlists/keyboard-patterns.txt`
  - `hashcat -m 10900 layer-07/hashes/PBKDF2.txt wordlists/keyboard-patterns.txt`

Layer 8

- To to crack some hashes to start with (plain rockyou.txt is usually a good starting place to determine some pattern), ran:
  - `hashcat -m 15100 layer-01/hashes/sha1crypt.txt rockyou.txt`
  - `hashcat -m 1800 layer-01/hashes/SHA512.txt rockyou.txt`
  - `hashcat -m 10900 layer-01/hashes/PBKDF2.txt rockyou.txt`
- Noted that all passwords seemed to be the same format as layer 6, so decided to use the same wordlist.
- Ran:
  - `hashcat -m 15100 layer-01/hashes/sha1crypt.txt rockyou-subset.txt`
  - `hashcat -m 1800 layer-01/hashes/SHA512.txt rockyou-subset.txt`
  - `hashcat -m 10900 layer-01/hashes/PBKDF2.txt rockyou-subset.txt`

Layer 9

- Downloaded human passwords version of crackstation.txt to start with.
- Ran:

    - `hashcat -m 15100 layer-07/hashes/sha1crypt.txt wordlists/human-crackstation.txt`
    - `hashcat -m 1800 layer-07/hashes/SHA512.txt wordlists/human-crackstation.txt`
    - `hashcat -m 10900 layer-07/hashes/PBKDF2.txt wordlists/human-crackstation.txt`
- Noted that:
    - All passwords seemed to be between 5 and 8 characters in length.
    - There were a few 7 or 8 digit passwords.
    - Perhaps we should move on to using the entire crackstation.txt wordlist.
- Cut crackstation.txt down to only digit passwords that are between 6 and 8 digits long:
    - Ran `awk 'length($0)>4' crackstation.txt > gt-5.txt` to get rid of passwords less than 5 characters long.
    - Ran `awk 'length($0)<9' gt-5.txt > lt-9.txt` to get rid of passwords more than 8 characters long.
    - Ran `grep '^[0-9][0-9]*$' lt-9.txt > wordlists/5-8-digits.txt` to get rid of passwords that were not made up entirely of digits.
- Ran:
    - `hashcat -m 15100 layer-07/hashes/sha1crypt.txt wordlists/5-8-digits.txt`
    - `hashcat -m 1800 layer-07/hashes/SHA512.txt wordlists/5-8-digits.txt`
    - `hashcat -m 10900 layer-07/hashes/PBKDF2.txt wordlists/5-8-digits.txt`
- Randomised the order crackstation.txt's contents and split it into multiple files to be run:
    - Ran `gshuf crackstation.txt > rand.txt` randomise the order of the contents of crackstation.txt.
    - Ran `split -l 10000000 -d --additional-suffix=rand.txt rand.txt file` to split the shuffled wordlist into separate files.
- For X in 0->4 ran `hashcat -m 10900 layer-07/hashes/PBKDF2.txt wordlists/file0X.txt`.

Layer 10

- Downloaded Charles Eliot Norton's translation of Dante's Inferno.
- Ran `echo "$(cat dantes-inferno.txt" | tr " " "\n" | sort | uniq > dante-wl.txt` to create a wordlist from the epic poem.
- Ran:
    - `hashcat -m 15100 layer-07/hashes/sha1crypt.txt wordlists/dante-wordlist.txt`
    - `hashcat -m 10900 layer-07/hashes/PBKDF2.txt wordlists/dante-wordlist.txt`

Module evaluation

What I liked

- It was interesting to get some experience of the world of hash cracking.
- It was good that practical assignment requirements were well documented and defined so that they could be graded with automated tests.

What I disliked

- I did not like that students were made to do assignments they were told would be graded but were subsequently not.
- I did not like that students who had the ability to pay more money for processing power could achieve higher marks if they spent more money, as this would cause other students to get lower marks. This seemed unjust to me.
- I did not like how the module seemed to differ greatly from the descriptor students had signed up for.
- I did not like how there were so few lecture slides.