

Job Application - Zac Kologlu z5257261

Categories:

- Technical skills
- Security mindset
- Working with others
- Self management
- Something Awesome

Technical Skills

I have completed every single extended activity throughout the term:

SQLi:

Level 1 Flag:

FLAG-07e[REDACTED]0814a101ddbf

✓ Correct!

Submit

Level 2 Flag:

FLAG-a1e8c4[REDACTED]4c953c16969d

✓ Correct!

Submit

Level 2 Harder Flag:

FLAG-3a2ebc[REDACTED]8802ddaf724e

✓ Correct!

Submit

Level 3 Flag:

FLAG-67c2be[REDACTED]5b8a3422dedb

✓ Correct!

Submit

XSS:

Level 1 Flag:

COMP6441{Bu7_[REDACTED].FI4G}

✓ Correct!

Submit

Level 2 Flag:

COMP6441{YoUr_[REDACTED].d4nger0us}

✓ Correct!

Submit

Level 3 Flag:

COMP6441{YoU_ _7ag???

✓ Correct!

Submit

Level 4 Flag:

COMP6441{AI1_ _p4yOu7???

✓ Correct!

Submit

Buffer Overflow:

Level 1 Flag:

COMP6841{Hey_ _it!}

✓ Correct!

Submit

Level 2 Flag:

COMP6841{Oh_ _now}

✓ Correct!

Submit

Level 3 Flag:

COMP6841{STRING: _ORK}

✓ Correct!

Submit

Format String Injection:

Special Username:

COM _IIN

✓ Correct!

Submit

Admin username:

COM[REDACTED]DMIN

✓ Correct!

Submit

Admin password

TO[REDACTED]ORD

✓ Correct!

Submit

What flag is given when you get the right answer?

EN[REDACTED]47

✓ Correct!

Submit

Reverse Engineering:

Flag:

You_[REDACTED]trings

✓ Correct!

Submit

Flag:

no_s[REDACTED]_time

✓ Correct!

Submit

Flag:

Pa[REDACTED]ck

✓ Correct!

Submit

Flag:

Clif[REDACTED]es!

✓ Correct!

Submit

Malware Analysis:

SHA256:

b467e19eb0[REDACTED]3088f471c1d15a9671fe479baea

✓ Correct!

Submit

DLL

W[REDACTED]DLL

✓ Correct!

Submit

IP Address

16[REDACTED]38

✓ Correct!

Submit

Furthermore, I found and exploited a completely unintended vulnerability in the XSS challenge, which broke all 4 challenges, and created an extremely detailed writeup of the exploit:

<https://www.openlearning.com/u/zackologlu-q5ua57/blog/20VPwOrHowILearnedToStopWorryingAndMakeMyOwnCtf/>

This also involved registering a cheap domain: <http://20v.pw/> which is still live.

I dockerized both the buffer overflow exercises and the format string injection exercises, as they were debian 32 bit executables, and the only available official option was to download a 3GB Ubuntu Virtual Machine, and installing VirtualBox on many platforms involves installing third party kernel drivers (!!!), which I definitely wasn't comfortable doing.

The docker images are all open source: <https://github.com/insou22/6841-bof-docker> , <https://github.com/insou22/6841-format-docker> , and I publicised them heavily so students would know the option was available through Openlearning comments and twitter posts retweeted on @comp6441: <https://twitter.com/insoudev>

I presented lightning talks and demonstrated my reverse engineering of the new standard Optus NBN router management protocol, which is...

tweeted here: <https://twitter.com/insoudev/status/1229589746759819264> ,

open sourced here: <https://github.com/insou22/optus-router-tools> ,

packaged here: <https://www.npmjs.com/package/optus-router-tools> ,

examples packaged here: <https://www.npmjs.com/package/optus-router-tools-examples>

My Something Awesome was extremely technical, all the details are outlined in my poster, paper, and source code - all (conveniently) located at <https://github.com/insou22/sa-obf> .

This involved working directly in JVM Bytecode (Java assembly) for the entire project, and even writing 300 lines of raw bytecode to create an extreme obfuscation transformation not yet seen in the wild involving onionising class files by decrypting and loading them inside one another at runtime, and discovering a transformation that breaks almost all decompilers involving finding a strong discrepancy between the rules of the class-file format (think ELF file for Java) and how the JVM interprets them, and the rules of Java and how decompilers attempt to reverse Local Variable Tables.

Security Mindset

I maintained a strong security mindset throughout the course, always looking out for potential vulnerabilities in my doings and attempting to see such things through as much as possible, even if the outlook is not looking very likely.

Optus-Router-Tools - When we were (forcibly) “upgraded” to the NBN, Optus provided a standard router for us to install on our network. Applying my security mindset, I realised that this router would be standard across a large portion of Australia, and any security analysis performed on it could prove to be extremely beneficial to the greater Australian population. Even though I’m only 18 and had barely started my first security course, I decided not to give up and to persevere as much as possible to learn the details of the management protocol, even through painfully heavily obfuscated javascript and hundreds of network requests to sift through. Eventually, I succeeded in my reverse engineering of the new standard Optus NBN router management protocol, and open-sourced all my discoveries:

<https://github.com/insou22/optus-router-tools>

<https://www.npmjs.com/package/optus-router-tools>

<https://www.npmjs.com/package/optus-router-tools-examples>

I also presented a lightning talk on the subject to my tutorial in the hopes of inspiring them to reverse-engineer their routers, and also contribute their findings to the community.

For my job application, I also decided to open-source a lot of my working during the router recon, which is now available here: <https://github.com/insou22/optus-router-recon>

The files of interest are router_recon_pretty.js and index.js

I also demonstrated a strong security mindset with the XSS challenges, in which I found a very small unintended bug, and against all odds pursued it as much as possible to see how big of a vulnerability I could turn it into, which as I was able to perform some extreme minification of some JavaScript, and was able to chain the vulnerability with a misconfiguration of the page’s same-origin policy, I was able to create arbitrary code execution in all of the challenges, proving just how easy it can be to accidentally create vulnerable software. This is all outlined in extreme detail here:

<https://www.openlearning.com/u/zackologlu-q5ua57/blog/20VPwOrHowILearnedToStopWorryingAndMakeMyOwnCtf/>

A (partially redacted) quote from Clifford Sesel on my reverse engineering challenge writeup:



Clifford Sesel 2 days ago



I like your workflow and writeups. Definitely the right mindset.

<https://www.openlearning.com/u/zackologlu-q5ua57/blog/ReverseEngineeringChallenges/>

Working with others

I have demonstrated multiple acts of community building, professional and ethical behaviour, etc.

I dockerized both the buffer overflow exercises and the format string injection exercises, as they were debian 32 bit executables, and the only available official option was to download a 3GB Ubuntu Virtual Machine, and installing VirtualBox on many platforms involves installing third party kernel drivers (!!!), which I (and I'm sure many others) definitely wasn't comfortable doing. The docker images were all open sourced: <https://github.com/insou22/6841-bof-docker> , <https://github.com/insou22/6841-format-docker> , and I publicised them heavily so students would know the option was available through Openlearning comments and twitter posts retweeted on @comp6441: <https://twitter.com/insoudev>

After I dockerized the buffer overflow challenges, the creator of the format string challenges actually made a callout which I responded to with the newly created dockerized version.

[Download challenge OVA here \(UNSW Login Required\)](#)

Apologies for the weird download method, there were some technical difficulties. I'm hoping someone will download the binaries from the challenges and create a Dockerfile version (Would be great evidence of *community*).

 caff - a month ago

Be the first to like this  Like  Subscribe  History  Sub-Pages

I demonstrated strong ethical behaviour when I found my vulnerability in the XSS challenges:

ZK

Zac Kologlu
Thu 27/02/2020 10:54 PM
Lachlan Jones

    ...

Hey caff, it's Zac from your class, the one who broke his home router and causes way too much trouble in your tutes.

I found a super cool ****unintended**** xss bug in your blog ctf that let me break into all the challenges instantly and get all the flags without anything special or changing anything.

I'm down to do a lightning talk on it because I think its pretty damn interesting, but i feel like im already doing too many with the 2 home router ones already planned.

Whaddaya reckon?

Also ill write a private blog post on the bug and email you a link to it when im done if youre interested.

Cheers, Zac.

ZK

Zac Kologlu
Fri 28/02/2020 1:49 AM
Lachlan Jones

    ...

Update: Finished the blog post: <https://www.openlearning.com/u/zackologlu-q5ua57/blog/20VPwOrHowILearnedToStopWorryingAndMakeMyOwnCtf/>

Reader's discretion is advised, it's 1500 words of post-midnight unedited garbage, so I wouldn't bother if you have anything important to do over the next 12 hours or so.

Otherwise, I think this is a relatively cool rabbit hole out of my random stuff over the years, so hopefully you find a bit of interest in it too.

Cheers.



Lachlan Jones

Fri 28/02/2020 8:14 AM

Zac Kologlu ✉

Hey Zac, can you give me permission to view the page please?

Thanks,
Lachlan



Zac Kologlu

Fri 28/02/2020 8:52 AM

Lachlan Jones ✉



you should be able to see it now, I thought you could see private posts but the only other options I can see is the whole course can see it which I was trying to avoid, but should be fine im sure.



Lachlan Jones

Fri 28/02/2020 8:57 AM

Zac Kologlu ✉



Good call, I'd rather the whole course not see it. I've had a chat to the challenge author who says yes it's possible, if unintended.

So basically - awesome work. Keeping it private it also great evidence for any Professionalism we ask you to show in the JobApplication. I'm seriously impressed.

Awesome work, love it, keep it up!

Lachlan



Zac Kologlu

Fri 28/02/2020 9:03 AM

Lachlan Jones ✉

Sweet, so I'll assume you've had a read now and I can set it back to private?

Cheers.



Lachlan Jones

Fri 28/02/2020 9:06 AM

Zac Kologlu ✉

I have, awesome write-up. Perfect. I seriously can't express how happy I was to read all of that.

Please set it back to private now.

Well done!

Lachlan



Zac Kologlu

Fri 28/02/2020 9:14 AM

Lachlan Jones ✉

No probs, back on private.

Cheers.

Furthermore, since taking COMP6841, I have been actively attempting to get my peers to try out some new languages, which provide better security and safety than something like C or C++ (as extremely strongly evidenced throughout the practical components of this course), for example:

- Rust - A language that can be used as a drop-in replacement for C/C++, which provides compile-time absolute memory safety (using a borrow-checker) - no use after frees, no memory leaks, no race conditions, no buffer overflows, no undefined behaviour, and all zero-cost abstractions with no runtime to provide execution speed on par with C/C++.
- Haskell - A pure functional programming language that encourages its users to think carefully about the programs they're constructing, and compose them in a way that very often limits the possibilities of obscure bugs in production, due to things such as its extremely powerful type system, and its functionally pure composition.

I would like to request evidence from everyone here that I have been attempting to convince people to move away from dangerous languages that are easy to create security vulnerabilities in such as C and C++, and to move to trying out much safer languages such as Rust and Haskell - please respond to this message if you support this claim.

erik



Yes can confirm you've been pushing rust for a year and a half now



But yes I confirm that you have been steering away from unsafe practices and languages

androo



I can confirm that you have pushed for safer languages like Rust and Haskell over ones like C/C++. You have particularly noted Rust's memory safety through features like its borrow checker.















All the time

Self-Management

I demonstrated professional and ethical behaviour through the xss exploit as outlined in the working with others section.

I completed almost all of the standard 6441 coursework...

▸ Getting Started	Completed: 4 of 4 
▸ Module 1	Completed: 4 of 5 
▸ Module 2	Completed: 6 of 6 
▸ Module 3	Completed: 4 of 4 
▸ Module 4	Completed: 3 of 3 
▸ Module 5	Completed: 4 of 4 
▸ Module 6	Completed: 0 of 0 
▸ Module 7	Completed: 3 of 4 
▸ Module 8	Completed: 2 of 3 
▸ Module 9	Completed: 2 of 2 
▸ Module 10	 Available Monday, 20th April (12:00am). 

And every single extended 6841 exercise, as proved in the technical skills section.

Although I wasn't able to complete every single exercise on time each week, I made sure to make an effort of going back and completing the vast majority of them, evidenced in the screenshot. This means I finished the course with 32 / 35 standard exercises completed, and 22 / 22 extended exercises, for a total completion rate of 54 / 57 exercises, or 95%. Although it is not perfect, I am very happy with how I was able to manage my time with my other courses (Extended Operating Systems, Extended Algorithms), teaching two COMP1511 classes, the move to online learning AND teaching with covid-19, working as a cloud software developer at The NRMA, while simultaneously almost getting stuck in Queensland for an extremely stressful two weeks in the middle of term when flights started getting shut down and borders were being closed.

I also engaged in extreme amounts of self-directed learning, as outlined and evidenced throughout every other section in the Job Application.

Although I missed a single tutorial during that two week period as a result of those external factors, during my otherwise perfect attendance throughout the term I was a keen participator in student discussion, and provided lots of feedback and discussion with other students on their ideas. Evidenced from email:

Evidence of COMP6841 claims



Lachlan Jones

Sat 18/04/2020 10:22 AM

Zac Kologlu ✉

Hi Zac,

I can confirm that you:

- a. Attended all tutorials
- b. were an active participant in both case study and general class discussion
- c. frequently both questioned and praised ideas raised in class

Yours in professional communication,
Lachlan

Something Awesome

My something awesome materials are located here:

Github repo: <https://github.com/insou22/sa-obf>

Poster: <https://github.com/insou22/sa-obf/blob/master/Something%20Awesome%20Poster.pdf>

Obfuscation paper: <https://github.com/insou22/sa-obf/blob/master/Obfuscation%20Paper.pdf>

Backup video: <https://youtu.be/m6vU0NUpbDo>

All source code located in repo.

My something awesome and related materials were all submitted on-time.

I presented my something awesome on April 16th to Lachlan's tutorial, where I demonstrated in a live demo:

Java source syntax, java compilation to class files, decompilation of normally compiled files, the transformations of my obfuscator applied to a normally compiled class file, and the effects of those transformations on two separate extremely popular decompilers, and how they were completely broken by my transformations. I also demonstrated that after the obfuscation, the pre-obfuscation and post-obfuscation binaries both executed identically as normal.

I then gave a quick outline of my advanced assembly encryption onion transformation, and displayed where to learn more about my project and read my papers - all in the allotted 2 minute time slot.