



Bauhinia Newsletter

Vol. 01 • July 2024

Vol. 01




Contents

- QR Code Recovery
- SECCON
- BinDiff
- Threat Analysis
- Kubernetes Networking
- Flipper Zero
- Generative Model
- Web Debugging Proxy
- LLL Primer
- DLL Sideloading
- and more...

Bauhinia
In oscon!

Table of Contents

Table of Contents	1
Foreword	2
Knowledge Sharing	
The journey of my first cybersecurity certificate – <i>Jimmy</i>	3
Reflections on “Reflections on Trusting Trust” – <i>Stdor</i>	4
The Power of QR code recovery – <i>a1668k</i>	5
Bauhinia CTF 2023 Image Factory – <i>cire meat pop</i>	6
TetCTF 2024 LordGPT: Microsoft Azure nOAuth Bug – <i>vow</i>	7
Binary Similarity: Overview of BinDiff – <i>wwkenwong</i>	8
How LLL works (simply) – <i>Eason</i>	9
Traffic Routing in Kubernetes – <i>ensy</i>	10
Flipper Zero 推坑簡介之 NFC – <i>Gon7K</i>	12
Threats Analysis for Running Out of Paper in Public Toilets – <i>apple</i>	14
Introduction to Generative Model – <i>streamline</i>	16
Interacting Breakpoints with OWASP ZAP API – <i>vikychoi</i>	18
APT techniques studying: DLL sideloading – <i>botton</i>	20
PuTTY’s P521 vulnerability, and a LLL primer – <i>Mystiz</i>	22
Events	
小妹火山遊  – <i>grhkm</i>	24
Onsite Hardware Problem in SECCON 2023 Final – <i>harrier</i>	26
SECCON Trip in Japan – <i>hoifanrd</i>	27
Ad-Hoc	
Isekai Tensei Hakka Vol 1 Issue 1 – <i>Ozetta</i>	28
Isekai Tensei Hakka Vol 1 Issue 2 – <i>Ozetta</i>	30
Credits and Afterwords	32

Foreword

Welcome to the first public edition of our newsletter, presented by Black Bauhinia (blackb6a) team members. Black Bauhinia is a Capture-the-Flag team from Hong Kong founded in 2019 and have been actively participating in CTF games since then. Whether you're an industry expert or a student, we hope this newsletter will inspire you about different aspects of CTF and the cybersecurity landscape.

What is CTF?

Capture The Flag (CTF) is a popular type of cybersecurity competition that challenges participants to solve various puzzles and problems to capture hidden "flags". Often, players are required to break a system and workaroud the security measure to get the flags.

CTFs are designed to simulate real-world cybersecurity scenarios, providing a platform for learning and demonstrating skills in a fun, competitive and legally safe environment.

About Black Bauhinia

Black Bauhinia is a CTF team from Hong Kong dedicated to advancing cybersecurity knowledge and skills. Our mission is to foster a community of learners and professionals who are passionate about cybersecurity and eager to tackle new challenges.

Black Bauhinia

Also known as


- BlackBauhinia
- blackb6a

Website: <https://b6a.black>

Twitter: blackb6a

[Sign in](#) to join the team.

Captured on 16 June, 2024



A team based in Hong Kong.
(We are NOT affiliated with any associations.)

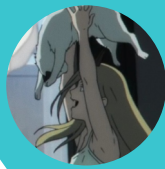
Participated in CTF events

2024 2023 2022 2021 2020 2019

Overall rating place: **24** with **327,686** pts in 2024

Country place: **1**

Looking forward, we are committed to continuing our efforts in organizing, participating CTFs and contributing to the cybersecurity community. We encourage newcomers to dive into the world of CTFs and experience the immense benefits they offer. We extend our heartfelt gratitude to our team members, participants, and supporters who make this newsletter. Join us in our journey, and let's capture the flag together!



The journey of my first security certificate

- Jimmy

Burp Suite Certified Practitioner (BSCP) is one of the cheapest cybersecurity certificates you would find on the market. It costs USD100 per attempt, and if you fail, you could retake the exam as many times as you want, if you can afford it. However, you will need Burp Suite Pro to take the exam, which costs an additional USD450 per year (but you could save the cost by using the 30-day trial version). The exam format is fun. You got 4 hours to hack 2 websites. You need to complete 3 stages one-by-one. First, compromise a user account. Next, utilize the user account to escalate privilege and log in as the admin. Finally, you use the admin account to read the secret located in `/home/carlos/secret`.

The scope of the exam is quite broad in my opinion. Port Swigger includes 30 categories of labs, which adds up to 264 labs in total. There are 3 difficulties: apprentice, practitioner, and expert. Most of the labs are well-guided, you could mostly finish them if you read the paragraphs. You are advised to do all labs of the apprentice and practitioner level. Did I follow the instructions? No, because I am a naughty hacker! That's why I failed my first attempt. But I did pass the second one easily after learning the lesson. Interestingly, the passing rate of the exam seems quite low. If you search for reviews, people usually take 3-4 attempts.

The most time-consuming part of the exam is perhaps "figuring out what's wrong". Therefore, Burp Suite got a perfect chance to promote its own Scanner and Collaborator architecture. And I am definitely impressed by it. Although it is imperfect, it makes me witness how powerful it can be in locating vulnerability. And I wonder if I could develop my own tools in doing similar things.

While the overall experience is quite fun, and I learned a lot, BSCP is perhaps one of the least known security certificates. The more famous OSCP/CISSP are still dominating the market. Therefore, I wouldn't recommend you to spend months completing all the labs, just to pass the exam. However, it would be a fruitful journey if you take advantage of the educational labs.



Burp Suite Certified Practitioner

This certification, created by PortSwigger's Web Security Academy, demonstrates that I have the ability to:

- Detect and prove the full business impact of a wide range of common web vulnerabilities.
- Adapt attack methods to bypass broken defences, using knowledge of fundamental web technologies.
- Quickly identify weak points within an attack surface, and perform out-of-band attacks to attack them.

PortSwigger are the creators of the Web Security Academy and Burp Suite, the world's leading toolkit for web security testing.





Reflections on “Reflections on Trusting Trust”

Stdor

Abstract We present a “trojan horse” (in python) based on the paper “Reflections on Trusting Trust”.

Ken Thompson, the lead designer and programmer of UNIX, really needs no introduction. Despite his contributions to the UNIX system and the C Programming Language, he decided to talk about security in his Turing Lecture “[Reflections on Trusting Trust](#)” when he won the Turing Award in 1983. It's now a required read in most undergraduate cybersecurity courses. The talk is significant in a sense that it introduces a fundamental concept in security - **supply chain attack** and the **root of trust**. I recommend reading the 3-page lecture first before trying this lab ([code available here](#)).

First we present a simple python “compiler” and “runtime”; implemented using marshal (a serializer) and compile. This is far from “real” compilation, but we make the same assumptions - the binary is relatively hard to audit, and the source code relatively easier.

Case I (Figure 3.1 in the original paper)

Here's the situation described in Figure 3.1 in the original paper - **using a benign system compiler to compile source code**. You have to compile the compiler yourself in the first command (which I indicated "please ignore"); because you would not run it if I supplied a compiler.bin directly to you (and you shouldn't). But let's assume that compiler.bin is provided to you by someone else (e.g., installed as system default), and is in fact benign. In this case, you get a benign compiler, you get to compile your program hello.py, everyone is happy.

Case II (Figure 3.2 in the original paper)

Now Ken suggests that the binary compiler could be backdoored. In our case, we'll “backdoor” python programs with a single line of print statement by adding a print("pwned") statement before compilation. It can be easily done, in compiler-evil.py. **We assume that we have access to compiler.bin only, but not its source code**.

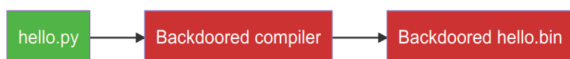


Fig 1. “Backdoored compiler” pollutes hello.bin. Green items are trusted items; red ones are malicious

This case is easy to detect. Suppose you ask for the source code of compiler.bin. If it turns out to be compiler-evil.py, you can spot the backdoor code.

Case III (Figure 3.3 in the original paper)

To make the backdoor harder to detect, **we backdoor the compiler binary used to compile a compiler**. Let's imagine the following. As an educated cybersecurity analyst, you don't trust the default compiler so you compile your own compiler. You have already audited the compiler source to have no backdoors (“Trusted Compiler Source”). But the compiler you used (“Backdoored compiler”) will compile itself (which is backdoored) instead when it knows it's compiling a compiler. At the end, you get a “Backdoored Compiler Binary” from “Trusted Compiler Source”.

The big picture looks like this. The right one is what you imagined, and the left one is the actual situation, namely, the “Trusted Compiler” is actually a “Backdoored compiler”. The backdoor propagates.

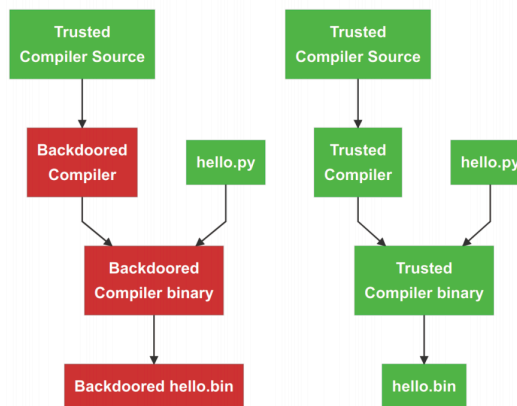


Fig 2. “Backdoored compiler” pollutes the compiler, then hello.bin

Despite having audited the compiler source code and compiling from there, your final binary is still backdoored. **You wrongly trusted the compiler that you used to compile your trusted compiler source**.

The moral is obvious. **Security is relative**. Behind every security claim, there's an ultimate root of trust. Here are some more lessons learnt:

- Source code-level audit gives you a false sense of security if you use an untrusted compiler to compile that source.
- When you claim a software is secure, what exactly are you saying? How can you be sure the whole “Silicon → Circuit → CPU → BIOS → OS → Compiler → Software” chain is not being tampered with?
- Spy chips can and indeed exist. But let's try not to be conspiracy theorists.

References. [Original Paper](#), [Slides](#), [Video on the paper](#), [Follow-up to the paper](#), [Quine Guid](#)



The Power of QR code recovery

By a1668k



In Firebird CTF 2024, just_lay and I have created a QR code recovery challenge 🐱 What really inspires us to create this challenge is because of this [Github write-up](#). At first glance, the given QR code has too many covered pixels, which seems to be impossible to repair. But with the **Reed-Solomon Decoder** in [QrazyBox](#), you can recover the flag by just plotting all the uncovered pixels, together with the fixed patterns (including the Finder, Timing, and Alignment patterns) 🐱 [P.S. Really don't know why it only got 2 solves??? I think this is an easy challenge...]

But why can we recover that many missing pixels? Isn't that QR codes only offer up to 30% recovery capacity with correction levels H? But our given QR code has more than 60% missing pixels... To understand why the recovery mechanism still works in our case, time to introduce the error-correcting codes used in QR codes - **Reed-Solomon error correction**.

QR codes use Reed-Solomon code for error correction, which the message will be encoded to Reed-Solomon blocks. With the normal error correction method, the highest error correction capability of a QR code is around 30% with an H level, which means it can recover at most 30% damaged bytes in each Reed-Solomon block.

However, there is a type of correction in Reed-Solomon code - **Erasure correction**, which can **double** its error correction capability if it is an **erasure code** - that is, if we know **where the errors are**, we can recover up to **60%** of the corrupted pixels! In addition to this, in larger QR symbols, the message is broken up into several Reed-Solomon code blocks and interleaved together, making the localized damage less likely to overwhelm the capacity of any single block.

In our case, as we know where all the missing pixels are, with the designed placement of the bobcats which makes sure that there are not too many missing erasures for each Reed-Solomon code block, we just have enough pixels to cover the whole QR code and get the flag! 🐱

I have deliberately avoided discussing the theory and implementation of Reed-Solomon codes in detail and skipped lots of details about QR code formatting here. For more details, here are some links for you to do further studies: (maybe it can create a good crypto challenge? Idk xDD)

https://en.wikipedia.org/wiki/QR_code

https://en.wikipedia.org/wiki/Reed-Solomon_error_correction

<https://merri.cx/qrazybox/help/extension-tools/reed-solomon-decoder.html>

Happy Hacking! 🐱



Bauhinia CTF 2023 Image Factory

cire meat pop

舊年我喺 Bauhinia CTF 出左一題 pwn 題叫 Image Factory，which 我個人認為系自己出過嘅所有 pwn 題裡面唯一一條稱得上“有翻咁上下難度”嘅 pwn 題，所以想喺呢度分享一下呢題玩咩同我嘅 intended solution。

TLDR，呢題 host 住一個 nc service，連過去之後你可以 input 任意 format 嘅圖然後個 program 就會 output 翻一個你指定嘅 vector image format (e.g. svg) 嘅圖比你。Attachment 入面有用嘅除左 Dockerfile 以外就只有 main.c。由於 main 見唔到明顯嘅bug，再者亦 Dockerfile 入面有特登 git clone 一個 external library — AutoTrace 落黎 build，正常人下一步就應該會開始研究呢個用黎 convert image 嘅 library。

今次呢條 pwn 嘅 intended solution 分別會用到 AutoTrace 嘅 info leak 0-day 加 libc arbitrary write 0-day。

info leak 0-day

TGA圖片儲存顏色嘅時候可以使用Indexed Color Mode，每個pixel唔直接儲存RGB值，轉而係儲存一個index。TGA嘅header會有一個color map data，pixel嘅RGB值會refer去color map data[index]。

input-tga.c:520

```
if (hdr->colorMapType == 1) {
    unsigned char *temp, *temp2, *temp3;
    unsigned char index;
    int xpos, ypos;

    temp2 = temp = image.bitmap;
    image.bitmap = temp3 =
        (unsigned char *)malloc(width * height * 3);

    for (ypos = 0; ypos < height; ypos++) {
        for (xpos = 0; xpos < width; xpos++) {
            index = *temp2++;
            *temp3++ = cmap[3 * index + 0];
            *temp3++ = cmap[3 * index + 1];
            *temp3++ = cmap[3 * index + 2];
        }
    }
    free(temp);
    free(cmap);
}
```

一個pixel嘅RGB值會refer到cmap[3*index]附近。由於index無做到checking，RGB最大可以refer到cmap[3*0xff]呢個位，但cmap嘅大小係由image header控制，所以user可以入一張cmap係16bytes chunk不過某個pixel嘅顏色係refer緊cmap[767]嘅圖，呢個好明顯係一個access out-of-bound vulnerability。

libc arbitrary write 0-day

input-bmp.c:606

```
if (bpp >= 16) { /* color image */
    XMALLOC(image, width * height * 3);
    if (masks[3].mask != 0) channels = 4;
    else channels = 3;
} else if (Grey) { /* Grey image */
    XMALLOC(image, width * height * 1);
    channels = 1;
} else { /* indexed image */
    XMALLOC(image, width * height * 1);
    channels = 1;
}
```

你可以喺 <http://bit.ly/48ZSHzN> 搵到呢篇文嘅Detail Version 同 Full Solve Script，不過係咁啱講下啦，solve script 裡面首先用 info leak 0-day 去 leak address，然後用 offset 計翻個 libc base address 出黎，最後就用 libc arbitrary write 0-day 去 overwrite libc 嘅內容（呢度我選擇左 house of apple 嘅方法）拎 shell。

image係用黎裝住所有pixel嘅chunk，大小最大為width*height*3，因為一個pixel可以有RGB呢3個唔同嘅data。但係根據code logic，channels係可以assign做4，換言之一個pixel係可以有RGBA4個唔同嘅data，width*height*3係裝唔曬嘅。

input-bmp.c:631

```
rowstride = width * channels;
ypos = height - 1;
switch (bpp) {
    case 32:
        // ...
        break;
    case 24:
        {
            while (ReadOK (fd, row_buf, rowbytes)) {
                temp = image + (ypos * rowstride);
                for (xpos = 0; xpos < width; ++xpos) {
                    *(temp++) = row_buf[xpos * 3 + 2];
                    *(temp++) = row_buf[xpos * 3 + 1];
                    *(temp++) = row_buf[xpos * 3];
                }
                if (ypos == 0) break;
                --ypos; /* next line */
            }
        }
        break;
}
```

parser會從source image度copy啲pixel去image呢個chunk度，而temp就指住黎緊會copy入去嘅位置。由於temp=image+(ypos*rowstride)，ypos*rowstride理應小於image嘅大小width*height*3。

ypos同rowstride嘅初始值分別係width * channels同height-1。所以ypos*rowstride嘅初始值係width*channels*(height-1)。仲記唔記得上面提過channels最大可以係4。因為4w(h-1) > 3wh，所以會出現“寫過界”嘅情況。咁就係一個write out-of-bound嘅vulnerability了。

而當malloc一個size有足夠地大嘅chunk嘅時候，咁malloc出黎嘅image下面嘅chunk就必定係dynamic libraries。利用呢點，pwner就可以做到libc arbitrary write。



TetCTF 2024 LordGPT: Microsoft Azure nOAuth Bug

- VOW

1 - Introduction

TetCTF 2024 had an interesting web (misc) challenge related to Microsoft's Azure AD. Here is a simple walkthrough of the challenge, how one can prevent this type of vulnerability, and my thoughts on this challenge.

2 - Walkthrough¹

We are given a webpage that allows us to sign in with a Microsoft account to access an AI chatbot; however, when we do so, we receive an error message saying that we do not have access permission. The trick here is to edit the tenant ID of the OAuth URL link² to {common}, which allows anyone with a Microsoft account to sign in.

Once we are in, we can do some prompting with the bot, and we can look at our profile. We learn that each profile and chat ID is uniquely generated using Snowflake ID³, and there exists an admin profile.

Snowflakes are 64 bits in binary and have three main parts: the time, a machine ID and a machine sequence number. The time is given by the chatbot, so we can figure out the set of machine IDs and sequences by generating and checking multiple snowflakes. After that, we can easily figure out the ID of the admin profile.



Snowflake ID structure

Looking at the admin profile, we can see that it contains an email address, but if we use this email to sign in, we find that this email does not exist. The last part of this challenge is to create a tenant user, modify its email to the admin's email, sign in with the user's email⁴, and just like that, we have taken over the admin's account and we can get the flag.

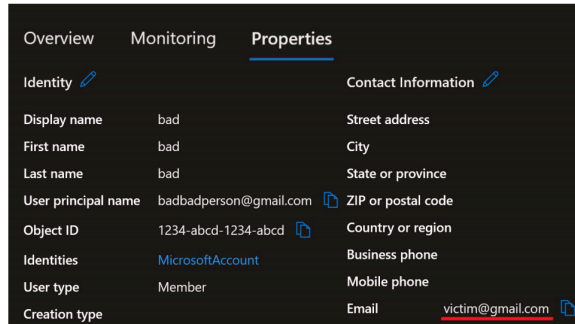
Flag: TetCTF{L0rdGPT1sTh3B3st!!!!}

¹ [My full writeup](#)

² [Microsoft identity platform and OAuth 2.0 authorization code flow - Microsoft](#)

³ [Snowflake ID - Wikipedia](#)

⁴ [nOAuth: How Microsoft OAuth Misconfiguration Can Lead to Full Account Takeover - descope](#)



nOAuth vulnerability, change user's email to admin's email

3 - Preventive measures

There are three bugs to this challenge, and there is something that can be done for each of them.

For the OAuth URL, this flaw would not happen if the application were configured to only allow users with permission to access it. For the Snowflake ID, one can simply implement server-side checking for cookies (which has not been imposed for profiles), and as for the nOAuth vulnerability, applications should not use email claims for authentication.



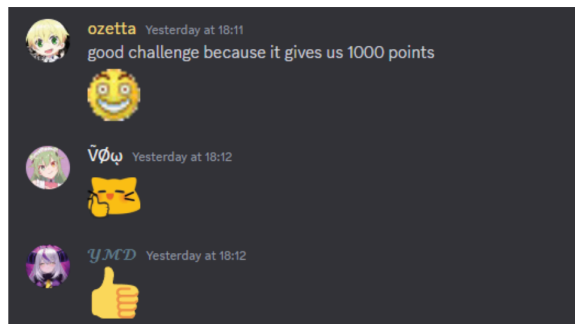
Configuration that prevents unauthorized accounts from signing in

4 - Thoughts

The vulnerabilities in this challenge are very interesting, but finding them out is insanely tuning. No source code was provided, and we literally had to guess the vulnerabilities (it took us 10 hours just to get onto the website, and we probably would not be able to solve the challenge if the author did not give out hints for each part).

5 - Conclusion

Misleading description, fake web/AI challenge, hohosiu Azure tuning challenge, however:



Points > Challenge Quality (x)

Special thanks to Ozetta and YMD for helping me solve/tune this challenge.



Binary Similarity: Overview of BinDiff

Author: Ken Wong

While binary comparison tools like BinDiff rarely come up in CTF, the topic of binary similarity (BCSA) is an important area of ongoing research in modern software security. Apart from vulnerability analysis, binary similarity techniques have various uses like identifying open source license violations in commercial software, malware analysis, etc.

Compared with source diffing, applying line-by-line diffing on assembly instructions can fail to detect functionally equivalent programs optimized differently or compiled with varying tools. For example, the same source code compiled at different optimization levels or with different compilers may exhibit major syntactic differences while behaving identically. Line-by-line matching cannot account for such variations. More advanced approaches leverage semantics, like symbolic execution, to check equivalence at the code behavior level while requiring heavyweight computation. BinDiff is the current industry-standard solution for comparing similarity between binaries and was engineered by Halvar Flake in 2003. It has served as the most practical solution in the industry since then. In this newsletter, I will first give a high-level overview on the design of the BinDiff.

While BinDiff was engineered in 2003, it remains the most practical solution in the industry. It first consumes disassembly code generated by reverse engineering platforms (like IDA Pro, Ghidra, and Binary Ninja), then compares functions in two binaries and derives a match between them based on scores. The core idea is to use three levels of statistical features to represent a function, namely the number of basic blocks, the number of edge blocks, and the function call graph. Based on the first two representations, BinDiff initially obtains a match on two sets of functions. Then, it iteratively refines the matching result with different features and strategies derived from the control flow graph (CFG) and the function call graph. A score is obtained from each of the strategies, and the final matching result is obtained based on a combination of the scores with the confidence of each of the strategies. Afterwards, the final match result is presented as two lists: a list of functions successfully associated with each other and a list of functions that could not be associated.

As a Computer Science student, you might already notice that deriving matching between two sets of CFGs is equivalent to solving the graph edit distance between the two sets, and is an NP-hard problem. Meanwhile, the design of BinDiff allows it to derive a matching result in a reasonable time. The major reason lies in the technical design and feature selection of the signature matching algorithm. The design of BinDiff uses a significant number of features present in the binary function to derive the similarity score. For example, it uses the function name (if symbols are available), the hash of the function's raw bytes, and the MD index (a graph hash function designed by Halvar Flake in [here](#) to support fast lookup during matching. It is based on the topological information of the input graph) of the function CFGs and call graphs.

Another technical hurdle faced in matching assembly instructions is register allocation. Register allocation is an important step performed by compilers during code generation. It involves mapping high-level variables and values to machine registers that will be used during program execution. This step may not generate consistent results across different versions of the compiler. The implication is that, if we use instruction as a unit of comparison, it leads to false negatives during matching. To address this and improve the performance in comparison with the CFGs, BinDiff uses the prime product of instruction mnemonics to represent an assembly instruction. These primes will then be multiplied to form a unique signature for the function.

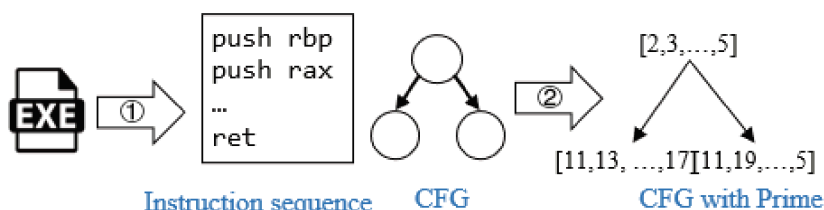


Fig 1 Overview in BinDiff preprocessing a binary function

To wrap up, the use of various graph heuristics is the key to the success of BinDiff in solving the binary function matching problem and remains the industry standard up till now. To further understand the design, the primary source would be the recent open-source repository of BinDiff, the two papers and blog post written by Thomas Dullien, as well as the publication from Zynamics. And since the BinDiff result is stored in SQLite format, examining the BinDiff result directly will provide insight into how matches were generated.



You might know many solution of crypto challenges are using lattice based attack such as LLL algorithm for RSA, ECDSA, etc. But do you know how it works?

Idea of lattice

We know that in linear algebra, a set \mathcal{B} of vectors is called a basis if the vectors are linearly independent. Then a lattice is formed, which is the set of all integer linear combination of vectors in the basis \mathcal{B} : $\mathcal{L} = \mathcal{L}(\mathcal{B}) = \{\sum_i a_i \vec{v}_i : a_i \in \mathbb{Z}\}$.

Moreover, there are 2 lattice based problems: shortest vector problem and closest vector problem, LLL algorithm can find the approximate solution of those problem. Therefore, in some situation, e.g. when the RSA private exponent d is too small, we are able to recover the secret key or message as long as we can construct a right basis and the solution contains those secret information.

But how can we convert a cryptographic setting into a lattice based problem or more specifically, transform a RSA equation into a basis?

Transformation

Assume we get this RSA equation: $a_2x^2 + a_1x + a_0 = r \pmod{n}$, also values of a_i , r , and n are given. we can build up a system of linear equation like $Ax = b$ and this:

$$\begin{bmatrix} 1 & & & & 0 \\ & 1 & & & 0 \\ & & 1 & & 0 \\ & & & 2^{xlen} & 0 \\ a_2 & a_1 & -n & a_0 - r & 0 \end{bmatrix} \begin{bmatrix} x^2 \\ x \\ k \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} x^2 \\ x \\ k \\ 2^{xlen} \\ 0 \end{bmatrix} \in \mathcal{L}, \text{ which } \mathcal{L} \text{ is over } \mathbb{Z}_n$$

2^{xlen} means value of x is at most $xlen$ -bit length, which is for limit the approximate value of x within 2^{xlen} after apply LLL algorithm.

The key part is the last row, after applying matrix multiplication, the Ax will be:

$$\begin{bmatrix} x^2 \\ x \\ k \\ 2^{xlen} \\ a_2x^2 + a_1x + a_0 - r - kn \end{bmatrix}$$

the equation of the last row is equal to $a_2x^2 + a_1x + a_0 = r \pmod{n}$, which k is a constant. After apply the LLL-algorithm on the basis matrix A , it becomes a LLL-reduced basis and a shortest vector in the lattice can be found in the first vector of the reduced basis.

You might also be confused why LLL-reduced basis can contain the solution like this. Simply put, the Gram-Schmidt orthogonalisation process used to compute the given basis into an orthogonal basis and then Lovász condition is used in LLL algorithm to help length reduction and ensure that the second vector should be not much shorter than the first.

And According to Minkowski's First Theorem, LLL algorithm is able to solve x when: $x \leq \sqrt{d} |\det(\mathcal{L})|^{1/d}$, which d is the degree of the polynomial (highest power of x)

This is a very rough explanation for LLL algorithm. In fact, most of the time we might meet more complex cryptographic setting, such as the equation can be multivariate polynomial. To solve those problems, we have to understand more maths and cryptography.

Finally, I want to give a special thanks to Cousin and hoifanrd for teaching me more about number theory and lattice-based cryptography.



Traffic Routing in Kubernetes

ensy

Background and Introduction

For the past year I've been trying to develop infrastructure to host OI/CTF problems, and recently I've been playing around with kubernetes and figured I should share some stuff I've discovered, particularly on the network side of things. Though I'm still learning how kubernetes works, I hope that through this article more can learn to route traffic in kubernetes!

Services

Before we get going with network stuff, let's first understand how kubernetes works.

To deploy a simple program, one creates a group of container(s) called a "pod", and runs the program inside it. These pods run on one or more servers, often spread across different locations, called "nodes".

Pods with the same label selector(defined in a configuration file), can be selected and grouped together, forming what is known as a "service", allowing one to load balance requests to pods and more.

Here are the different types of services one can choose from:

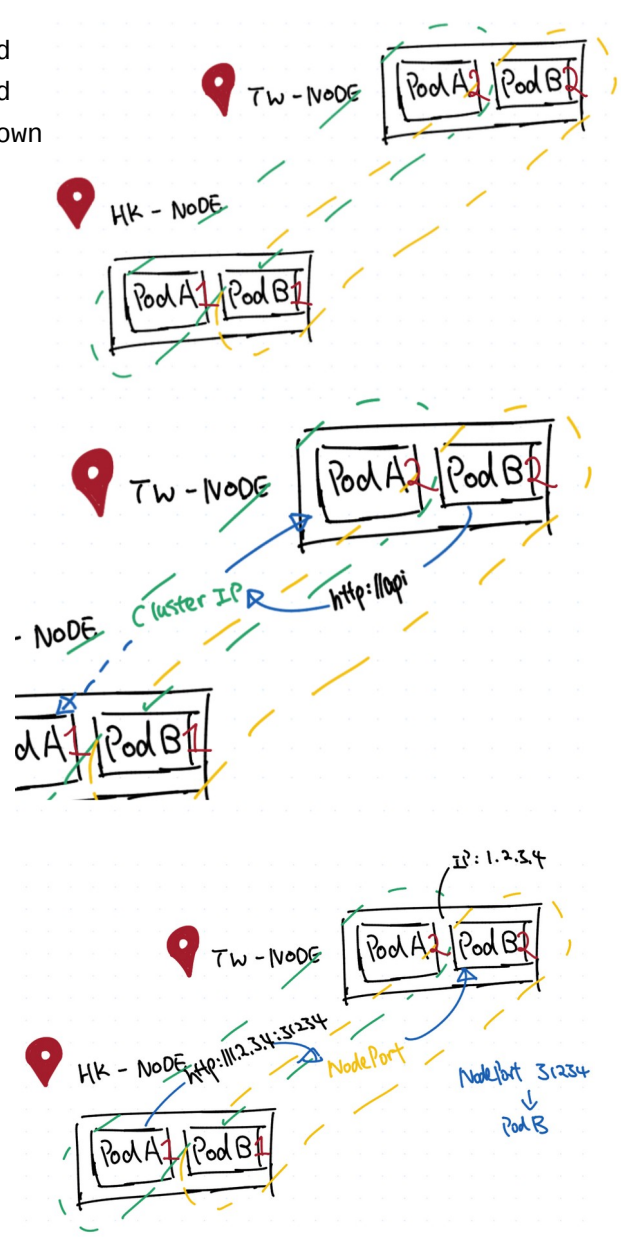
ClusterIP

ClusterIps exposes pods to the internal kubernetes network, for example, you can use a ClusterIp service for database pods, as they should not be publicly accessible.

Additionally, one can also expose ClusterIps using an Ingress.

NodePort

A NodePort service exposes a port[30000-32767] on your nodes(can be chosen with label selector)(each node should have different ips, so you access the service with http://nodeip:portyouchose). NodePort alone is very troublesome to work with, because you'll have to maintain a list of ip addresses of the nodes, which may scale up or down.



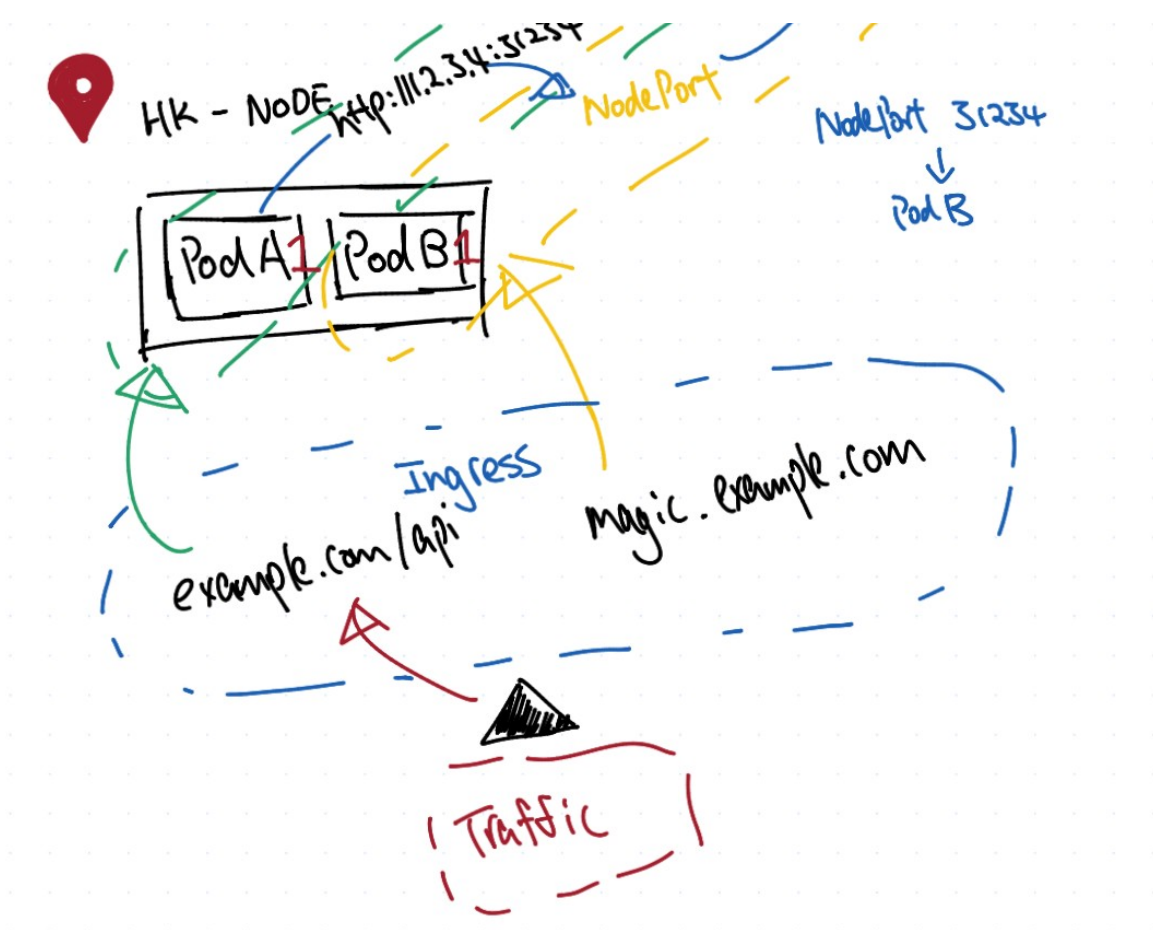
Load Balancer

A load balancer service is built on top of the NodePort service, but adds load balancing functionality to distribute traffic between nodes. The service creates an external network infrastructure to direct network requests to pods in the cluster. Cloud platforms that host kubernetes usually offer their own load balancer services to manage external load balancing, which is often pricey and hosted outside of k8s. Load Balancer services have their own unique, publicly accessible IP address that clients can use to connect to.

External client -> Loadbalancer(http://x.x.x.x) -> Worker node IP -> NodePort -> ClusterIP Service -> Pod.

Ingress

Though not a service, it can act as a router in front of your services, exposing them to the outside world with subdomain based or path based routing, and can be accessed via a public IP address just like a Load Balancer.





Flipper Zero 推坑簡介之 NFC

話說我買咗 Flipper Zero 咁耐都未認真用過, 要不是 Mystiz 邀請寫文我都未認真睇有咩玩。(大誤)

簡單講 Flipper Zero 就係一把 Hardware Hacking 嘅萬用刀, 雖然每樣功能都未必係最優秀, 但係細細隻係會你想拎出街把玩嘅玩具。今期主要會簡單介紹下當中功能。

係半隻手板咁大嘅 Flipper Zero 可以支援下面嘅功能:

- NFC (High Frequency Card)
- RFID (Low Frequency Card)
- Sub-1 GHz Transceiver
- Infrared Transmitter
- GPIO Pin
- iButton
- 仲有內置小工具

Firmware Flashing

係玩 Flipper Zero 之前, 我會建議先 Flash 咗個 Firmware 佢。Flipper Zero 原生嘅 Firmware 上面 Block 咗某啲功能, 例如 Sub-1 GHz 某啲 Frequency 嘅傳輸。如果想 Unlock 所有 Feature 的話可以考慮 [Xtreme Firmware](#)¹ 或者 [Unleashed Firmware](#)², 兩款firmware 除咗解鎖咗 feature 之外, 仲會包埋其他 developers 嘅 plugin 入去, 集各家之大成。

甚至覺得唔夠的話可以睇埋 GitHub [all-the-plugins](#)³, 下載自己適用嘅 plugin。

註: Flipper Zero plugin 需要 compile 同 Firmware 夾 version, 未必一 down 就用得。

註2: 係我交咗文之後幾日, XFW 嘅主要 contributor 出走自己整咗 [Momentum FM](#)⁴, 都幾好用。

NFC

宜家坊間一般嘅門禁卡都會用 High-Frequency Card, 當中最弱嘅係 Mifare Classic 1k (絕對唔係講緊你的悠遊卡)。

Mifare Card 平常嘅攻擊手法有:

- Brute-force Attack (Nested Attack)
- Sniff Key (Darkside Attack)

Nested Attack

雖然 Mifare Classic 1k 有 16 個 encrypted sectors, 拜佢用 LFSR 所賜只要解鎖到其中一個 就可以 Unlock 所有 Sector。(見 LFSR State Rollback)

而有啲廠商偷懶嘅原因用 default key or simple key 可以直接係 [NFC keys](#)⁵ 搵到。甚至我宜家用嘅 XFW Firmware 自己就包咗 3941 條 key。只要 User 爆到其中一個 sector 一條 Key 就可以了。

Darkside Attack

如果咁唔好彩 Key 唔係 Dictionary 入面, 只能用 Sniff 嘅方法 Read Key。Mifare standard 之中 Reader respond 會包含 4 bits of keystream, 導致 Mifare card 上面嘅 key 可以解譯並截獲。不過 Flipper Zero 只能 Fuzz and Emulate card ID 去嘗試 trigger Reader Response, 唔似 iCopyX 咁直接 intercept reader 同 card 之間嘅 communication。

註: Flipper Zero 都做到 Darkside attack, 使用 Detect Reader read 咗讀卡器嘅 Nonce 用 Mfkey32 解 key。



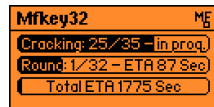
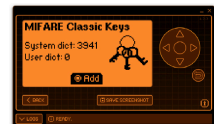
Flipper Zero 係咪 Tama-gotchi? 的確無錯。



Kickstarter 時代先有嘅黑色版本, 屬尊爵不凡嘅 nuttyshell 所擁有。



新台幣 329 嘅悠遊卡, Pika Pika



借 Pentest 名義去 Crack Key

¹ <https://flipper-xtreme.me/>

² <https://flipperunleashed.com/>

³ <https://github.com/xMasterX/all-the-plugins>

⁴ <https://momentum-fw.dev/>

⁵ https://github.com/Stepzor11/NFC_keys/blob/main/mf_classic_dict.nfc

RFID & Others

至於 Low Frequency Card (RFID), 基本上用 Wiegand format 無做 encryption, 直接 read 完 emulate 就可以, decrypt 都怪返。

當 Flipper Zero 成功 decrypt 卡片之後, 就可以生產出一個 dump file, 用於之後 emulation。而且唔一定係 Mifare Classic 先支援, Amiibo 嘅 NTAG 215 都適用。

香港八達通係用 FeliCa (NFC-F), 而台灣 iCash 2.0 用 Mifare DESFire, Flipper Zero 完全無反應。如果要針對 NFC craft attack, 直接上 ProxMark 3 更划算。我嘗試用 Flipper Zero 去讀 iCash 2.0 卡片, 但係卡住 read 唔到的。

But, 當我拎出 iCopyXS (Compatible with ProxMark3) 之後一睇就知道係咩 standard, 證明 Flipper Zero 距離 NFC hacking tools 仲有一段距離。現階段真係一隻好玩嘅玩具, 只能欺負啲 less secure 嘅 system。

HITCON 2023 嘅禮包之一有一張 CUID card 嘅物體, 當時官方話你知道係咩就好有用但係無點講清楚。CUID 其實係一張可以重複擦寫所有 sector 嘅 Mifare Classic 1k card, 比一般嘅 UID card 更難發現佢嘅重複擦寫特性。因為有啲 Reader 會嘗試係咁卡嘅時候係 Sector 0 寫入資料, 可以寫入的話會 reject access。CUID 就係防止 detection 之下嘅產物。配合 Flipper Zero 可以係 Flipper Zero 有 Card dump 要 physical 插卡情況之下 access system。雖然講到好強大但係遇到 online system 時候 NFC 只會得返 Authentication 功能, 失去 storage 嘅用途。

Update on 16 March 2023:

本來諗住係 UMC 卡過 data 返 CUID 發現卡關, 如果你打算 clone 卡請務必睇咗先。Mifare Classic original card 坊間叫 M1 卡, 通常係廠出預 read-only。後來有啲人動歪心思做啲 writeable 嘅 blank card, 統稱叫 magic card。Magic Card Gen 1 又叫 UID card, 的確好用但係可以 write 所有 sector。之後又出咗 magic card Gen 2 (aka CUID card)。iCopyX 係可以直接 erase and write UID 同 CUID card。隨住時代進步不滿足於 Mifare Classic, 有人研發咗 Ultimate Magic Card (UMC), 可以 emulate 埋 Ultralight / NTAG, 多咗功能仲好啱可以 clone 埋 NAMCO 張 Ultralight。⁶ BUT ! iCopyX 未用 new PM3 client, 只能正確辨認 UID 同 CUID。而 Flipper Zero 只支援 UID 同 UMC, 雖然萬用但係成本高不少:

UMC :>200 HKD

CUID / UID: <1 HKD



同埋 PM3 同 Flipper Zero NFC dump format 有少少唔同要自己轉換。

註: 學費馬話齋今期 newsletter 啲位已經用完, 要留返下次先可以 share 點做 conversion。

預告 (如果有下期)

就會講下 Flipper Zero 其他功能, 好似點用 Infrared 功能係 OWASP Meetup 嘅時候操弄 Speaker 個 Projector 咁。亦會講下 iOS 17 嘅 BLE crash 點樣用 Flipper Zero 做 PoC。

註: 我完場先玩嘅, 沒有一個 Speaker 受到傷害。



私貨乃木坂 iCash 2.0
但係畢業得七七八八



iCopyXS 高光時刻



HITCON 送嘅 CUID
再買要課金 100 新台幣



紅色框係 Projector Menu 被我
撇咗出嚟

⁶ https://github.com/RfidResearchGroup/proxmark3/blob/master/doc/magic_cards_notes.md



Threats Analysis for Running Out of Paper in Public Toilets

Running out of paper (ROP) in a toilet is one of the most undesirable outcomes to happen when using a toilet. In [previous literature](#), Youtuber @lambdatech concludes that an RFID-based door lock system can prevent ROP under all circumstances. However, the video failed to comprehensively analyze the threat landscape associated with ROP, overlooking several crucial aspects like supply chain and contamination, thus rendering its conclusion unreliable. In this article, we will perform a threat modeling for toilet paper in public toilets, focusing on ROP. We will also review existing ROP prevention controls and products worldwide.

This article attempts to answer the 4 key questions for threat modeling of toilet paper:

- **What are we working on?**
Asset profile of toilet paper (TP)
- **What can go wrong?**
Threats leading to ROP
- **What are we going to do about it?**
Controls for ROP mitigation
- **Did we do a good enough job?**
Existing controls for ROP resilience

The security objective of this article is to mitigate possible threats leading to ROP situations, meaning that “no toilet paper could be used”. Other undesirable outcomes for toilet usage are not a focus here. We would consider a public toilet in a shopping mall when considering the threats they are facing.

Asset profile of toilet paper (TP)

The first step in understanding the possible threats for ROP is identifying assets. We would do so by examining a typical use flow of TP in a toilet to understand its intended usage.

Use Flow: Enter the toilet cubicle, Defecate, Take around 10 ~ 100 cm of TP, Wipe, Flush and exit.

Janitors are tasked to refill TP periodically: Purchase toilet paper from retail, Store it in a storage room, and periodically check and refill the TP holder when empty.

With the above information, we can create an Asset Profile as follows:

- **Asset:** Toilet paper
- **Asset container:** TP holder, Storage Room
- **Actor (Users):** Toilet user (Outsider), Janitor (Insider)
- **Access:** Toilet users should only take an *appropriate amount* of TP from the holder’s dispenser. The janitor should refill the TP from the storage room to the TP holder by opening its cover.

Also, the TP must fulfill some CIA requirements to be usable:

- **Integrity:** TP should be clean, dry, and suitable for wiping sensitive skin.
- **Availability:** TP should be available *all the time* in the toilet when the mall operates. It should be *long enough* for wiping.

For simplicity, we would focus on the TP itself and ignore the threats to the asset container (e.g., the TP holder). The TP supply chain would also affect its integrity and availability. Trees, water, and electricity are required to produce TP, and their integrity/availability would affect TP production. We would not analyze the risk of supply chain attack in this article.

Threats leading to ROP

We can consider “running out of paper” as the outcome of two kinds of risks:

1. TP used up. There is no TP in the TP holder.
2. TP is unusable, such as being unclean, wet, i.e. contaminated.

There are many outside or inside threats that could result in the ROP situation, and we can identify them by constructing a threat tree (attack tree) by considering the following factors:



- **Actor:** Inside (I), Outside (O)
- **Motive:** Accidental (A), Deliberate (D)
- **Outcome:** Disclosure, Modification, Interruption, Destruction/Loss

For simplicity, we would only consider human threats to the system. Natural threats, such as humidity in spring that causes TP to wet, would be omitted from the analysis.

With the threat tree, four main threats leading to ROP could be identified:

- [I,O/A,D/Destruction] **TP Contaminated** during usage or cleaning
- [I,O/D/Modification] **TP Stolen** from TP holder or storage by user or janitor (insider attack)
- [O/A/Interruption] **TP Used up** during usage
- [O/A/Interruption] **TP Out of stock** in retail

Controls for ROP mitigation

With the risks and threats identified, we can consider the appropriate technical, physical, or administrative controls to apply. Typically, the controls fall into the following categories:

- **Preventive:** Avoid or deter undesirable events from occurring
- **Detective:** Identify undesirable events that have occurred
- **Corrective:** Correct undesirable events that have occurred, or recover from undesirable state
- **Compensation:** Provide alternative solutions

The following table summarizes the applicable controls:

	Contamination	Used up	Out of stock	Stealing
Preventive	Fully covered TP holder Moisture / dust proof storage room	Multiple rolls of TP to achieve high availability	Prepare enough stock in storage to avoid regional or global ROP	TP holder / Storage room with locks Signage to discourage stealing TP
Detective	Check if TP are contaminated periodically	Check if TP available periodically Install sensors / CCTV for ROP / stealing detection Audit trail for TP usage (detect insider attacks, too)		
Corrective	Add signage with telephone number of the facility / call button to call someone to bring TP when ROP. Refill or replace TP when ROP			
Compensation	Encourage wiping using hand by installing soap. Install washlet (smart toilet seat) Purchase alternative TP, e.g. bamboo-based TP			

Existing controls for ROP resilience

Let us look at controls and products around the world for ROP resilience.

Preventive - Used up	Preventive - Stealing
	
Toilet in ICHIRAN (— 蘭) equipped with multiple TP in HA setup	Toilet in China equipped with facial recognition to prevent abuse

Preventive - Stealing	Preventive
	
Deterrent signage for toilet paper stealing in Japan. It says “Famous to be stolen - Toilet paper roll - 50 yen each”	A commercial TP holder that mitigates ROP from 3 threats: achieved HA to prevent use up, spare TP fully covered from contamination, and locked to prevent stealing

Introduction to Generative Model

The diffusion generative model is a class of latent variable models inspired by considerations from nonequilibrium thermodynamics.

(<https://arxiv.org/abs/2006.11239v2>)

It is in a high level, generating noise in latent space (known as latent noise), then use pretrained node to level by level, denoise the image into existence with the text prompt user input.

The pretrained model is obtained by training from the original dataset of image+ captions(text) and with large amount of compute resources, linking text concepts with image presentation, in this way, when not over-fitted, would be able to correspond to the user input to give image output that relies onto the text description.

A standard sd1.5 structure in a high level can be divided into below:

clip → diffusion pipeline(unet) → vae → output

clip	breaking down user prompt input into chunks of tensor data that gets deliver into diffusion pipeline
diffusion pipeline	where the noise image is generated and the denoise process is conducted (upon into latent space, which is still not final output that human can understand)
vae	converting latent space into pixel space (simply think of it as decoding the encryption of your message in a messaging app), which means the final output of an image

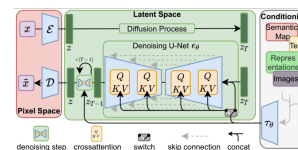
In a standard ai image generation, users might use stuff such as controlnet, animatediff, or other plugins/models that affect the final output, most of these are effective to the diffusion pipeline during the denoise process of images.

The older Deepfake setup is no longer available as the original roop plugin have since been unsupported by newer version of sd webui, and would require a rewrite that im lazy to do, here is a new setup that might require some modifying of the plugin to effectively do for video (it would be updated later so there is a batch process inpaint mask generating in segment anything tool, which allow batch upload of video frame+grounding box base on prompt, all automated):

Use forge version of a1111 webui, in parent folder of a1111 webui, do as follows:

```
git remote add forge https://github.com/llyasviel/stable-diffusion-webui-forge
git branch llyasviel/main
git checkout llyasviel/main
git fetch forge
git branch -u forge/main
git pull
```

To update the a1111 into forge version, then download the plugin from <https://github.com/pkulyiv2015/multidiffusion-upscaler-for-automatic1111> in extension tab, enable and reboot the webui, download photomakerv1.bin model from <https://huggingface.co/TencentARC/PhotoMaker/tree/main> obtain inpaint mask from <https://huggingface.co/TencentARC/PhotoMaker/tree/main>, put it inside the /models/controlnet folder



the image structure is from before SD3, applying to sd1.5, sd2.1, sdxl

Pseudo code example of how a diffusion model works:

```
# CLIP is a model / function that
transform your text prompt into a
tensor embedding
# textual embedding is a float vector
that machine can understand
context = CLIP(text prompt)
```

```
# initial noise latent that will be
denoised later
x = noise_generator(seed)
```

```
# ControlNet preprocessor, example:
line art preprocessor converts a
color image to a line art image
cond_hint =
preprocessor(conditional_input_image)
```

```
for i in range(steps): # denoise N
steps
```

```
# ControlNet is a model /
function that can map cond hint image
into cond hint latent for each latent
output of hidden layers
```

```
cond_hint_latent =
ControlNet(cond_hint, x,
timesteps[i], context)
```

```
# UNet is a model / function that
can predict how much noise is added
to the output image
```

```
# given input noise, noise level
and textual embedding
predicted_noise = UNet(x,
timesteps[i], context,
cond_hint_latent)
# sigmas[i] is a float that
represents the noise level, related
to the timesteps[i]
x = x - sigmas[i] *
predicted_noise
```

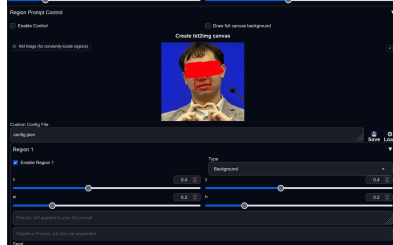
```
# VAE is a decoder that can convert
latent space to pixel space
output_image = VAE(x)
```

```
def UNet(self, x, timestep, context,
cond_hint_latent):
x = self.input_blocks(x,
timestep, context)
x = self.middle_block(x,
timestep, context) +
cond_hint_latent.pop()
for output_block in
self.output_blocks:
x = output_block(x, timestep,
context) + cond_hint_latent.pop()
return x
```

Disclaimer: we have obtained the agreement from the two human models in the following demonstration for educational purpose,do not attempt to do the following setup if you have not obtained permission from the person you try to do face swap to/from

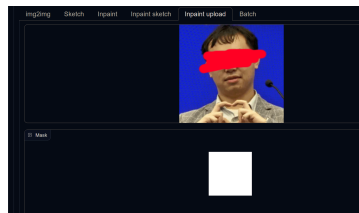
in the tiled diffusion tab, do the following:

enable region 1,drag the box until it fit the face,scroll down and click generate mask and save the mask



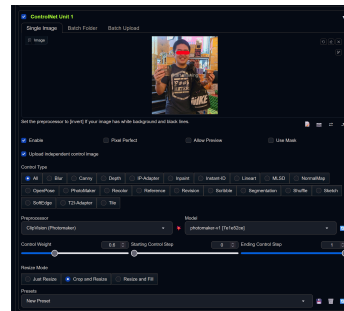
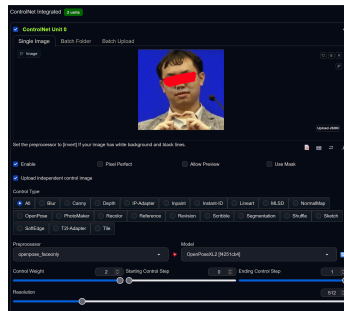
Save Region as a Mask

in the image to image tab,under inpaint upload, upload as follows:

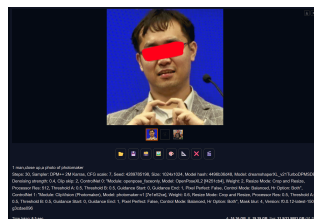


leave everything as default,set dimension as you pleased (multiply of 8)

denoise from 0.4~0.6, enable controlnet as follows:



photomaker are the one for the face you want to swap to,input the prompt with least amount of description such as a man,wearing glasses,close up, add a prompt of “a photo of photomaker” at the end then generate:



Hence the output, original pics(redlined their eye due to request from the models):



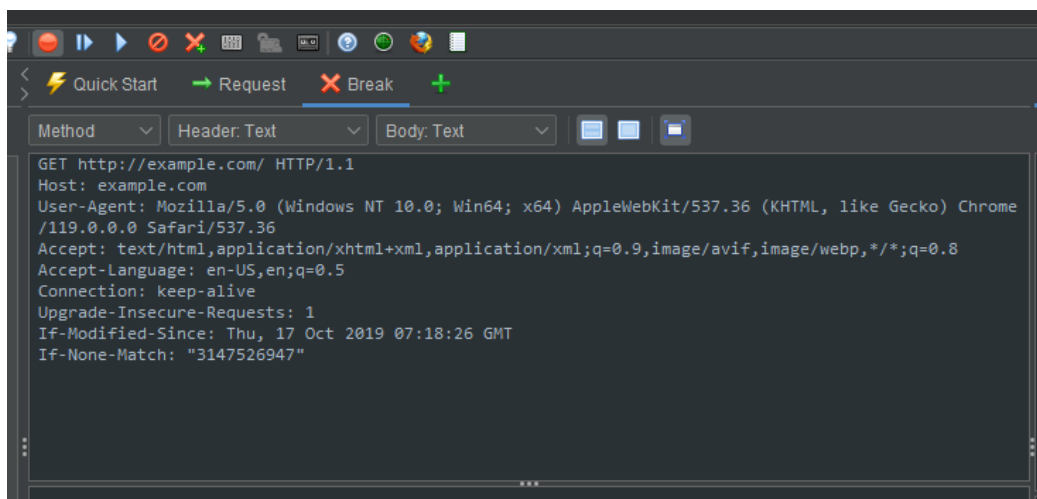





Intro

OWASP ZAP is an attack proxy which is capable of intercepting HTTPS traffic, modifying and replaying HTTP requests. It is a freeware that is highly comparable in terms of functionality with Burp, a commercial product that is the go-to proxy in the industry of penetration testing. Considering that I rarely find good materials that share some tips and tricks on ZAP's less well-known features, I would like to introduce you to the API interface of ZAP and share one of its relevant use cases in real world testing.

Breakpoints

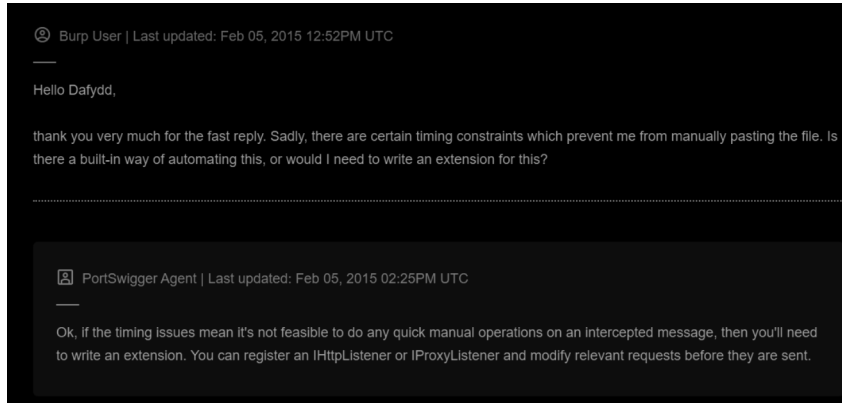
Breakpoints allows you to intercept a request from your browser and to change it before it is submitted to the web application you are testing. Let me share with you what it could do.



1. A break on all requests and response could be achieved by toggling the  button.
2. You can also edit the request directly by simply clicking and typing on the request header or body in the Break tab.
3. Clicking on the step button  would submit the request. The proxy will step to the response of the previous request.
4. If you clicked on the continue button , it would allow all requests and responses to pass through until the next breakpoint is hit.

Use Case

Why would you need the API when there is a great GUI easily available? Consider the following scenario that I found on the PortSwagger forum that would also apply on ZAP.



So long story short, ZAP API could be the solution for tampering requests with sophisticated payloads under time constraint. IMO, It is less laborious than implementing a burp extension.

To implement a script to solve the simplest case of tampering a specific request, add a HTTP breakpoint, loop to check if the desired breakpoint was hit, and modify the request. Forward the request and disable the breakpoints. This could all be achieved with a simple python script that interacts with the relevant APIs. Checkout the following link if you like code.

https://github.com/vikychoi/code_share/blob/main/demo.py

Real World Example

Mobile applications that utilize eKYC would usually send multiple images to the backend server to verify the validity of the credentials and liveness of the human image. By swapping images or videos in the request body with something like AI generated documents or faces, you might be able to achieve spoofing. Usually these applications would usually be hardened with encryption.. You could tamper the relevant encrypted HTTP messages via the ZAP API easily once you figured out the encryption scheme.





APT techniques studying: DLL sideloading

bottom

DLL Sideloading is a technique that exploits the **Dynamic Link Libraries (DLL)** search order in Windows to execute malicious code under a legitimate Portable Executable (PE) file.

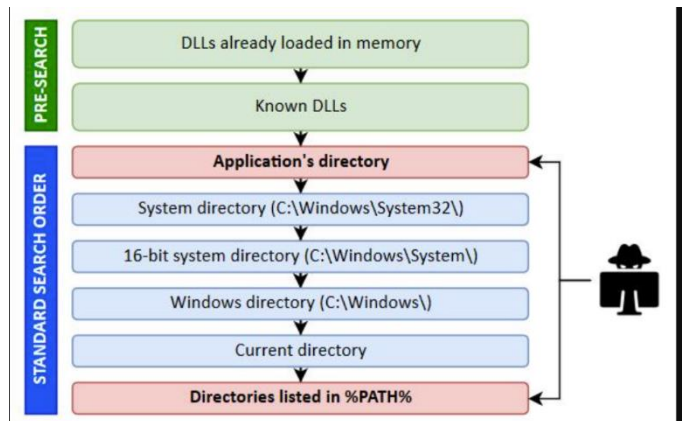
There are three primary methods through which the DLL Sideloading technique can be abused:

1. Code Execution on DllMain: By manipulating the DllMain function, attackers can achieve code execution when the DLL is loaded.
2. Code Execution on DLL Exported Function: Attackers can target specific exported functions within the DLL to execute their malicious code upon invocation.
3. DLL Proxying: This method involves replacing a legitimate DLL with a malicious one and redirecting the application's calls to the compromised DLL, allowing the attacker to execute unauthorized actions.

When a PE file attempts to utilize a function from a Dynamic Link Library (DLL) file, the DLL file is loaded either at link time or runtime using the **LoadLibrary** or **LoadLibraryEx** functions.

DLL Search Order

Developers have the option to specify an absolute path or just a filename when referencing the DLL file, particularly when they are uncertain of the exact installation directory. If the full path of the DLL file is not provided, the system follows a predetermined search order to locate and resolve the path. Unfortunately, this search order behavior can be exploited by attackers to execute malicious code within a seemingly legitimate PE file to carry out malicious activities such as code execution, persistence, or lateral movement by dropping a malicious DLL file in the application directory.



Code Execution on DllMain

The **DllMain** function serves as the entry point when a process loads a DLL file. Typically, we place our code within the **DLL_PROCESS_ATTACH** case, which ensures the code is executed as soon as the DLL is loaded. However, it's worth noting that this may not always be the case. Some programs may not invoke the entry function because the function call might have been prepared in advance using **GetProcAddress**, or deadlock issues may arise when creating thread tasks.

When you call **CreateThread**, a kernel thread object is created and scheduled. Once the thread gets a chance to run, the kernel calls all the **DllMain** functions with the **DLL_THREAD_ATTACH** code. Once that's done, the thread's entry point is called.

Code Execution on DLL exported Function

Exported functions from a DLL can be used to avoid the issues mentioned earlier. These functions can be obtained using tools like **PEBear** or **Dllviewer**. The names of the exported functions are hardcoded in the symbol table of the DLL file, allowing us to retrieve all the exported functions from it.

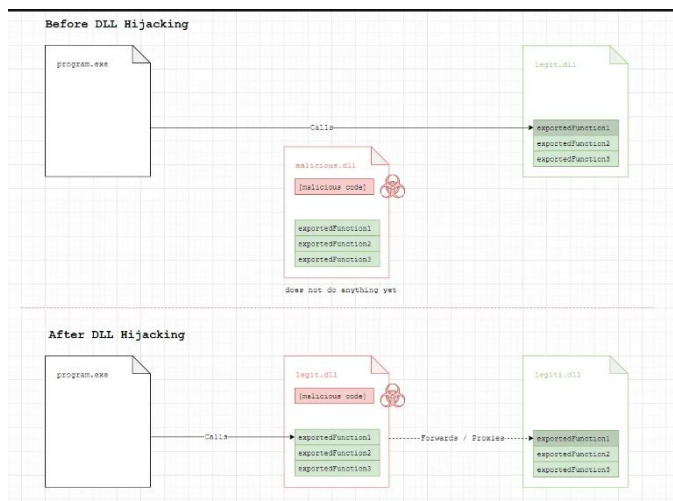
Exported Functions [322 entries]					
Offset	Ordinal	Function RVA	Name RVA	Name	For
1C7E0	7C	1E6E8	1E6D2	AppCacheDeleteIEGroup	
1C7E4	7D	1E732	1E71A	AppCacheDuplicateHandle	
1C7E8	7E	1E777	1E766	AppCacheFinalize	
1C7EC	7F	1E7BD	1E7A4	AppCacheFreeDownloadList	
1C7F0	80	1E808	1E7F2	AppCacheFreeGroupList	
1C7F4	81	1E84E	1E83A	AppCacheFreeESpace	
1C7F8	82	1E890	1E87E	AppCacheFreeSpace	
1C7FC	83	1E8D6	1E8BE	AppCacheGetDownloadList	
1C800	84	1E921	1E90A	AppCacheGetFallbackUrl	
1C804	85	1E969	1E954	AppCacheGetGroupList	
1C808	86	1E9B1	1E99A	AppCacheGetIEGroupList	
1C80C	87	1E9F4	1E9E4	AppCacheGetInfo	
1C810	88	1EA37	1EA20	AppCacheGetManifestUrl	
1C814	89	1EA79	1EA6A	AppCacheLookup	
1C818	8A	1EAB9	1EAA4	CommitUrlCacheEntryA	
1C81C	8B	1EB08	1EAA	CommitUrlCacheEntryBinaryBlob	

To execute our own code within a DLL, we can compile a DLL file with the exact same function name and place our code inside that function. Then, we can rename our compiled DLL file to the name of the DLL file we want to sideload. For example, if we want to sideload `msteamsupdate.exe` using `wininet.dll`, we would replace the original `wininet.dll` in the same directory with our modified DLL. When `msteamsupdate.exe` is launched and calls the exported function from `wininet.dll`, our code will be executed.

By following this approach, we can take advantage of DLL sideloading to inject our code into a target executable, leveraging the exported functions of a DLL to achieve our desired outcome.

DLL Proxying

DLL Proxying is another technique of exploiting the DLL search order. This approach is a bit more sophisticated, as it involves creating a malicious DLL that mimics the legitimate DLL. Unlike the previous methods, this does not solely rely on executing code when the DLL is loaded, but rather takes advantage of the specific functions that the application calls from the DLL.



In a DLL Proxying scenario, an attacker would create a malicious DLL with the same name as the legitimate DLL that the application is expecting to load. The malicious DLL would include the same exported functions as the genuine DLL, essentially 'proxying' these function calls to the genuine DLL. However, the attacker can insert malicious code into these functions, which will be executed when the application calls them.

This method requires a more in-depth knowledge of the legitimate DLL's

functionality, but it allows for a higher degree of control and stealth, as the application continues to function normally while the malicious code is being executed.

`Spartacus` is a useful tool which associated with `ghidra` script to help us automatically to generate the proxying function. You may checkout the details on the below for the usage of `Spartacus`

<https://www.pavel.gr/blog/dll-hijacking-using-spartacus-outside-of-dllmain>



PuTTY's P521 vulnerability, and a LLL primer

Mystiz

What PuTTY vulnerability? The developers released PuTTY 0.81 on April 15, 2024, urging their users to revoke the elliptic curve ECDSA private keys related to P521. It is announced that the vulnerability, assigned CVE-2024-31497, would make a P521 private key recoverable by an adversary if they have collected enough signatures from the victim. Additionally, the vulnerable function was in the code base on PuTTY 0.68 up to 0.80¹. I was pretty interested in how the bug was engineered, so I dug into their codebase. This is a snippet from their recent commit message², where the vulnerability is fixed:

As far as I know, the structure of our scheme is still perfectly fine, in terms of what data gets hashed, how many times, and how the hash output is converted into a nonce. But the weak spot is the choice of hash function: inside our `dsa_gen_k()` function, we generate 512 bits of random data using SHA-512, and then reduce that to the output range by modular reduction, regardless of what signature algorithm we're generating a nonce for.

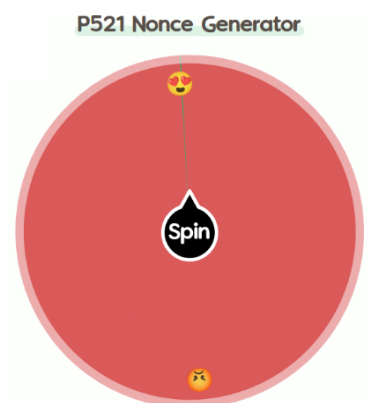
Their change in `ecc-ssh.c`, where the once vulnerable ECDSA signing function was at, showed that they replaced `dsa_gen_k()` with the newly implemented `rfc6979()`. Interestingly, ECDSA key recovery with predictable or biased nonce is not a rare topic in CTFs. There are plenty of ECDSA key-recovery challenges like *Flag is Win* in TSG CTF 2021, *Sign Wars* in SECCON CTF 2021 and *Signature* in TSJ CTF 2022³. @Merricx_ from X also mentioned⁴ that they wrote the exact problem like the PuTTY one last year.

The ECDSA basics, in P521-SHA256. Let's get started with the signing algorithm with P521-SHA256. P521 is an elliptic curve⁵, where you have a preset curve point G and an order n , which is of 521 bits.

1. Calculate $z = \text{SHA256}(m)$.
2. Select a cryptographically secure random integer k (the nonce) from $[1, n - 1]$.
3. Calculate the curve point $(x_1, y_1) = k \cdot G$.
4. Calculate $r = x_1 \bmod n$ and $s = k^{-1} (z + rd_A) \bmod n$. If $r = 0$ or $s = 0$, go back to step 2.
5. The signature is the pair (r, s) .

It is fine if you are not familiar with elliptic curves. We don't need to know what an elliptic curve is to understand the attack - knowing some modulo arithmetic would suffice. It should be hard for attackers to compute (r, s) from m without the private key d_A , or to recover d_A from (m, r, s) . However, it is not the case when they made a mistake, for instance, the k s generated are not random enough.

How is their k generated? PuTTY was using the `dsa_gen_k()` function⁶ which derives the nonce, k , with $k = \text{SHA512}(\text{SHA512}(d_A) \parallel \text{SHA160}(m))$. The PuTTY contributor mentioned that "This number is 512 bits long, so reducing it mod n won't be noticeably non-uniform⁷". While this is true for P256 and P384, it is not the case for P521. With that, k is sampled from $[1, 2^{512} - 1]$ instead of $[1, n - 1]$, which is much smaller than it intended to be. The resulting k is very biased - its top 9 bits are always zero. If we are spinning the wheel of fortune (the nonce generator on the right), is so biased that we are getting 🎯 every single time⁸.



¹ <https://www.chiark.greenend.org.uk/~sgtatham/putty/wishlist/vuln-p521-bias.html>

² <https://git.tartarus.org/?p=simon/putty.git;a=commitdiff;h=c193fe9848f50a88a4089aac647fecc31ae96d27>

³ *Signature* is a particularly good one. Go have a look if you are interested!

⁴ https://twitter.com/Merricx_/status/1780156751398293933

⁵ <https://neuromancer.sk/std/nist/P-521>

⁶ <https://git.tartarus.org/?p=simon/putty.git;a=blob;f=crypto/dsa.c;h=71fcd94a#l343>

⁷ PuTTY used q instead of n in their codebase. Changing the variable names for consistency.

⁸ If the wheel is fair, the probability of getting a 🎯 would be $1/512$.

How do we exploit the biased nonce for the private key? Assume that we have multiple message-signature pairs, i.e., we have $(m_1, r_1, s_1), (m_2, r_2, s_2), \dots, (m_N, r_N, s_N)$ such that

$$s_i k_i \equiv z_i + r_i d_A \pmod{n}, \text{ for } i = 1, 2, \dots, N. \quad [†]$$

[†] comes from step 4 of the algorithm with terms rearranged. In the above equation, we have n (the curve order, provided as a parameter of P521), z_i (the message digests), r_i and s_i (the signature). What is kept unknown are the k_1, k_2, \dots, k_N and d_A . The size of the unknown is about $512N + 521$ bits. On the other hand, since the congruence is in modulo n , we would say that we have $521N$ bits of information. When $N \geq 58$, we have more information than unknown⁹. We then can theoretically recover the k_i 's and d_A . The LLL algorithm is a good mathematical tool for finding *biased* (or *short*) integral solutions of equations, and we will be recovering the secret key with it.

Let's suppose that z_i' and r_i' are integers such that $z_i' s_i \equiv z_i \pmod{n}$ and $r_i' s_i \equiv r_i \pmod{n}$. [†] can be written as $k_i \equiv z_i' + r_i' d_A \pmod{n}$, which can be transformed to $k_i = z_i' + r_i' d_A + x_i n$ with x_i is an integer.

$$\begin{array}{cccccc} 2^9 k_1 & 2^9 k_2 & \dots & 2^9 k_N & 2^{521} & d_A \\ & & & \uparrow & & \\ \left[\begin{array}{cccccc} 2^9 z_1' & 2^9 z_2' & \dots & 2^9 z_N' & 2^{521} & \\ 2^9 r_1' & 2^9 r_2' & \dots & 2^9 r_N' & & 1 \\ 2^9 n & & & & & \\ & 2^9 n & & & & \\ & & \ddots & & & \\ & & & 2^9 n & & \end{array} \right] & \begin{array}{c} 1 \\ d_A \\ ? \\ ? \\ \vdots \\ ? \end{array} \end{array}$$

We will use the above lattice (matrix with integers) to find short vectors using the Lenstra–Lenstra–Lovász (LLL) algorithm. The algorithm finds an equivalent basis where the vectors are generally shorter. For simplicity, we will treat LLL as a black box¹⁰ – What we need to know is there is a `.LLL()` method in *Sagemath*. Anyway, we still have to learn how the lattice is constructed.

This is how I understand lattice reduction: The numbers on the right are the *hidden integral inputs* that will be decided by the LLL, such that the numbers on the top are their *outputs*. To be exact, the output of an entry is the dot product of the respective column vector and the input vector. Also, the shorter the target vector is, the more likely that the vector shows up after LLL. Let's use $N = 100$ as an example¹¹: The norm of the vector $[2^9 k_1, 2^9 k_2, \dots, 2^9 k_N, 2^{521}, d_A]$ is around $2^{523.50}$, while the norms of the remaining vectors range between $2^{527.30}$ and $2^{528.10}$, about 14 to 25 times as large as the target vector. In theory, we will get k_1 as the first entry because $2^9 k_1 = 2^9 z_1' \cdot 1 + 2^9 r_1' \cdot d_A + 2^9 n \cdot x_1 + 0 \cdot x_2 + \dots + 0 \cdot x_N$. The last two entries would respectively be

$$\begin{aligned} 2^{521} &= 2^{521} \cdot 1 + 0 \cdot d_A + 0 \cdot x_1 + 0 \cdot x_2 + \dots + 0 \cdot x_N, \text{ and} \\ d_A &= 0 \cdot 1 + 1 \cdot d_A + 0 \cdot x_1 + 0 \cdot x_2 + \dots + 0 \cdot x_N. \end{aligned}$$

After all, LLL returns a basis (a bunch of vectors). If one of them being $[v_1, v_2, \dots, v_N, v_{N+1}, v_{N+2}]$ happens to satisfy $v_{N+1} = 2^{521}$, we will be quite confident that the v_{N+2} is the secret key we want.

Lesson learned: Don't mess up 512 with 521... Well, who invented the P521 to confuse people?

⁹ Because $521 \cdot 58 = 30218 > 30217 = 512 \cdot 58 + 521$. The link from footnote 1 suggested that we needed $N \geq 60$. My experiment with Sage's default LLL setting needed $N \geq 90$.

¹⁰ I don't know LLL's internals either. Learn from Eason from another page in this newsletter, or from Cousin if you are really curious what it is: <https://www.klww.co/maths-in-crypto/lattice-based-cryptography-basics/>

¹¹ <https://gist.github.com/samueltangz/f7bf0dbfbbf01fa4a09dde8e58b3bfe1>



小妹^{isogeny}火山遊

講明先，呢篇嘢係我極度sleep-deprived嘅狀態下打出黎 😊 同埋有位 **anonymous reviewer** 叫我寫少啲廢話，所以我已經cut咗technical嘢，如果唔係應該會十六版紙 :D



俾少少background info/glossary啲唔熟crypto嘅人：

- ★ Sage: 一個Python嘅maths algebra library (+ preparser)，喺crypto ctf界尤其常見
- ★ Isogeny: (Post-quantum) Crypto嘅其中一個分支，同elliptic curve有少少關係
- ★ y7, jack, hellman, drago1729: 一堆ctf界嘅crypto神仙
- ★ ^^^ + Wouter, Luca, Jonathan: 一堆crypto界嘅神仙

話說舊年十二月見到個叫i-sage-ny days(= isogeny + Sage [3]) workshop，喺比利時嘅KU Leuven度搞，見個名幾有趣就reg咗(是真正理由)。個workshop雖然主打Sage，但係我reg完問吓先發現去嘅其實大部分都係crypto人。小妹呢啲small potato之前連professor都無見過咁濟(其實連lecture hell都未見過)，更何況係呢啲SOTA嘅大佬聚會，真是十萬分期待！原本諗住聖誕假補返啲paper同impl返個Wouter-Castrick attack，但係到依家都未impl 😊

Day 0: 好耐無試過朝早八點起身 😊 搭火車搭飛機搭巴士，眨吓眼就到咗。因為之前旅行食M記食到厭，所以今次有好乖咁上網睇咗食乜好，嗰晚就去咗間(望落)比較傳統嘅比利時餐廳 @ Brussels喫飯。間嘢個layout幾有趣，成件餐廳淨係得一個餐牌，寫咗喺塊「黑板」上面，每次有食客入嚟嘅時候，佢就會碌塊黑板過嚟你前面俾你睇，超有趣！我嗌咗個Carbonnades à la bière好似係啲啤酒燴牛肉or something... 我都唔係好清楚，不過真係好好味！啲牛一啲都唔韌又入味，同個薯蓉好夾，即使我平時mamaday薯X嘅食物都覺得呢餐好正。當然，我亦都嗌咗杯比利時啤酒Blanche de Namur，係比較清嘅麥酒 🍺，啱小妹飲~



★ ★ ★ ★ ★ Not posted

Rooms is 0. There's a god damn spider web next to my bed wtf? The only good thing is it's relatively convenient and also the cheapest I can find (apart from a youth hostel. Should've stayed there!!) please don't come here you WILL regret it.

跟住搭火車返酒店 @

Leuven，本身淨係上booking.com sort by price，諗住今次益自己少少，唔住hostel走咗去住間cheap酒店。只可以話好後悔 😊 事後睇返Google reviews得個2.8* (review唔係星數)，所以我都留咗個review。

Day 1: 第一日朝早首先係個歡迎talk、簡單嘅Sage tutorial同介紹點contribute去Sage (review workflow etc.)。我到嗰陣已經有唔少人到咗傾緊計，但係我一個人都唔識，想搵jack同y7 佢哋又認唔到人，就有啲尷尬咁縮咗喺角落頭 :” 最後好似有四十幾五十人，比想像中多！過咗陣俾talk嗰陣，見到坐我隔離嗰個出去present我先發現原來佢就係jack，再隔離就係y7，再再後嚟發現咗佢哋前面正正係hellman 😊 中間有個比較有趣嘅interlude係開頭Luca問在座邊個做過Sage dev，結果得五個人舉手，Luca就話”Okay there are five of you, and there is at least one I don't know...”之後我feel到全間房嘅人都望嚟住我(我:???? 小妹怕醜)就簡單做咗個self-intro，勁好笑又緊張。晏晝開始打code算係比較uneventful嘅時段。我哋分咗幾間房，有elliptic curve、hyperelliptic curve同quaternion algebra，我淨係睇得明第一個就去咗嗰度幫手。啲organisers建議我哋兩個兩個一齊pair programming咁寫code，我就同jack一齊搞CM method(佢講idea我搵code)。我哋Zulip上面有堆tracking issues，例如y7 suggest咗幾個project，而我自己都搞咗咗最簡單嗰兩個，算係唔錯嘅成績 ❤️

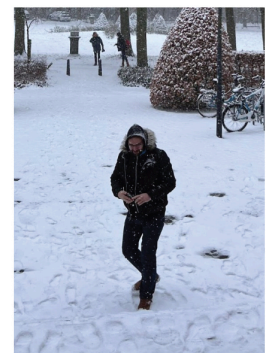
Is anyone here yet

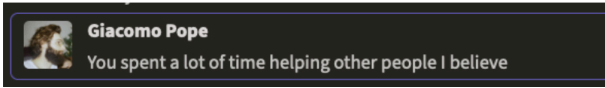
Update I'm at the table next to y7 and jack (edited)



夜晚我哋去咗間campus嘅酒吧飲酒，佢哋勁好笑喺度玩酒game，規則係如果講任何一個Silverman入面定義嘅字就要飲。同埋小妹唔識飲啤酒，y7就推介咗啲酒俾我飲，飲完先知原來比利時嘅啤酒通常比較濃，大概7%左右 😊

右邊附上早餐同 Hellman Snowman 🧑‍🎨 嘅圖，雖然white balance炒咗咗。





Day 2: 第二朝一開始我哋每組各自匯報吓progress。由於第一日唔係太多時間寫code，加上大部分人唔係太熟點樣做open-sourced development (例如fork同整branch)，所以整體唔算太多進展。同上一日一樣，我

哋同返各自嘅partner搵code。另外嗰日識咗位臺灣人 Lai Yi Fu (有睇過isogeny paper嘅可能會見過佢)。佢個Sage又係用唔到，所以我幫佢install返，但係搞極都搞唔到 😞 加埋好似無咗三個鐘 😊 究竟啲project幾時先唔會再用GNU Make...

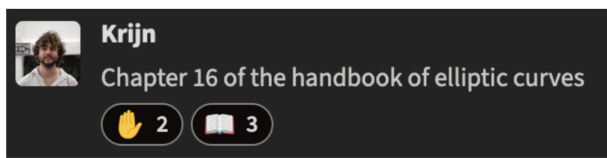
第二晚我哋食完飯之後(P.S. 有中餐食，surprisingly好味!)啲大大就半casually咁傾關於isogeny嘅嘢，我就負責做啦啦隊 + 得閒問吓啲trivial question。我諗我成個trip學關於isogeny最多嘅嘢就係呢晚，因為即使我未聽得明晒全部內容，但係聽吓聽吓啲neuron就開始連到。而且我覺得咁樣同聽lecture有啲相似：我以前都覺得上堂上lecture係毫無得益，但係依家我領悟到上lecture個重點唔應該係個內容本身，而係個lecturer點樣呈現個內容，因為一個(好嘅)lecturer往往會俾到好嘅motivation同intuition你，咁樣對學成個topic係十分有幫助。當然，前提係要有個唔係照稿讀嘅lecturer，而isageny嘅各位當然唔係咁嘅人。呢晚另一個印象深刻嘅位係去咗同y7一齊寫code。真心喺佢真人隔離望住佢搵code係好有趣嘅事，例如我發現佢同我一樣都係用Vim(智者之選 >>>> Emacs >>>>>>> Rest)，又見到佢成個寫code -> 試code -> 改code個過程等等。我最大收穫係發現原來IPython入面撇F2會彈個editor出嚟，而撇Alt + Return可以即刻run成段code唔洗落到最



```
rm -rf ./ * && git add . && git commit -m 'remove all failing doctests' && git push -f
```

尾嗰行，我諗淨係呢兩個shortcut喺過去兩個禮拜已經慳咗我幾十次x幾秒嘅時間。多謝y7! ❤️ Anyways其實我唔係好記得第二日晏晝做過啲咩，所以等我嚟問吓我partner啦! Oh...

Day 3: 嚟到最後一日啦! 首先有啲低能嘅係我前一晚要搞assignment + 其他嘢遲咗訓，搞到訓過龍，所以成個朝頭早嘅summary就係喺我睡夢中渡過 😊 我哋晏晝食完飯之後原來有個seminar，由Jonathan講關於一個叫做"Quaternion Embedding Problem"嘅問題。由於我對quaternion幾乎完全無認識，所以我完全聽唔明 😊 不過我覺得Jonathan present得幾好，即使我完全唔知啲formula講緊乜，不過都明到個high-level idea。另外我見到Christophe Petit出現，佢原本喺我附近嘅大學做research，我有諗過PhD去跟佢，不過佢依家好似走咗去法國 😞 我法國法國真係好多勁人...



(懷疑寫article可以reduce去bin packing，所以係NP-hard)

其實仲有好多嘢可以share，好多on9 bug，不過都係下次先... 呢個post已經過晒limit。最後留一個簡單exercise俾大家：請計算下面個S嘅generating function (as a graded ring)，之後deduce一條elliptic curve嘅defining equation出嚟 😊 Merci d'avoir lu! 898 ❤️



$$(p-2+1)=2p=\dim(S_{2p}); \text{ hence we have}$$

$$S_{2p}^+=(S_2)^p, \quad S_{2p}^-=(S_2)^{p-2} \vartheta_{00} \vartheta_{01} \vartheta_{10} \vartheta_{11}$$

for every $p \geq 2$. Suppose next that k is odd, say $k=2p+1$. Then, in the same way as above, we have

$$S_{2p+1}^+=(S_2)^p \vartheta_{00}, \quad S_{2p+1}^-=(S_2)^{p-1} \vartheta_{01} \vartheta_{10} \vartheta_{11}$$

for every $p \geq 1$. In particular, we have

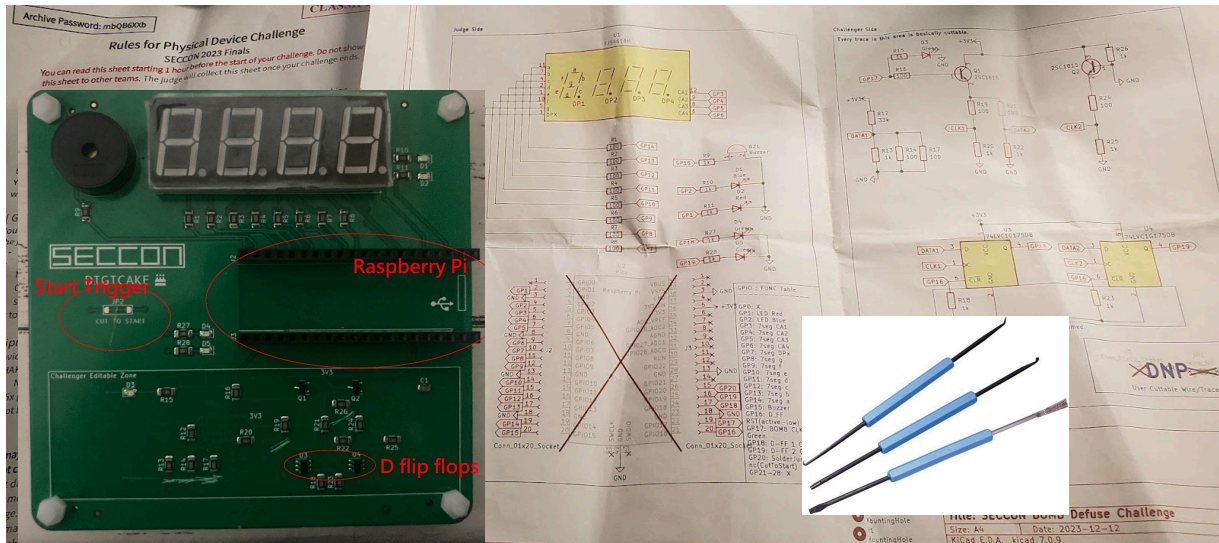
$$S=\mathbf{C}[x, y, z],$$

in which $x=\vartheta_{00}$, $y=(\vartheta_{11})^2$, $z=\vartheta_{01} \vartheta_{10} \vartheta_{11}$.



Onsite Hardware Problem in SECCON 2023 Finals

Onsite Hardware Problem in SECCON 2023 Finals by harrier



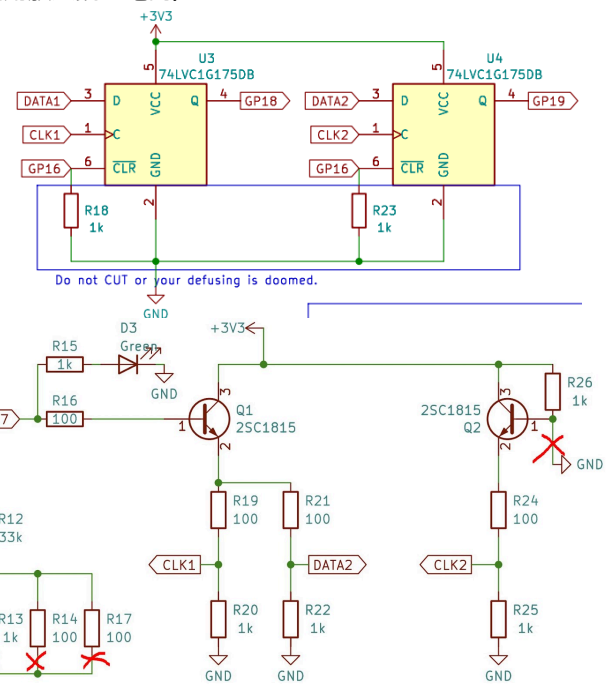
今次SECCON Final 想分享下條幾有趣既onsite hardware題目 digicake：佢係一條time attack 限時一個鐘既題目。限時既方法係咁：佢係platform 上邊有一個encrypted zip 檔（同埋一個好廢既故事背景description），每隊都有一個自己揀既時段（隊隊唔同），喺開始嗰時會收到一個circuit diagram，一張rule，zip 個password（入邊係raspberry pi 既 source）同埋一個實物（aka 電路版）既圖。（塊版係我地實際玩完塊版，留意有個“Challenger Editable Zone” 同啲我地attempt 既挫痕）

呢個時候我地先知要做啲咩：一個鐘後佢就會叫你入一間房做只得一次“defuse” 個電路版既attempt；入邊剩係有三支細「挫」俾你係指定範圍入面挫斷啲電線（仲有個無用既電路用放大鏡 + 電筒）。

我地要係限時內根據資料黎搵到成功 defuse 到既方法。

個challenge本身都幾簡單，target 係令兩個d flip flop 係十分鐘內 set (aka value = 1) = defuse
D flip flop 既mechanism係，當CLOCK (CLK) 係raising edge (由0變1 果下) 佢個output (Q) 就會sync DATA pin 既value。所以我地個目標係令DATA1 同DATA2 都set (即係high voltage)，從而去set 到兩個d flip flop。

DATA1 只要cut 斷佢同下邊三個resistor 既wiring 就可以令佢變 high (本身current 會由幾個resister 經 ground 流走，唔會去左個D flip flop 到)
DATA2 就比較麻煩，由於個CLK係由GP17 derive 出黎既，基本上佢既value 同CLK1 係一模一樣，會alternating，我地無方法可以用cut線既方法黎令佢永遠set。再睇CLK2，上邊有個transistor (Q2)，transistor 就好似一個電掣咁，佢限制電流唔可以由(3)去(2)通過，只有(1)通電 (high signal) 既時候，電流先可以由(3) 去(2)。由於Q1既(1)係駁住GND，佢一直都係 0V，頂既3.3V (+3V3) 就落唔到去個CLK2 到，所以CLK2 一直都係零。要SET到第二個Q flip flop，我地就要係DATA2 係 SET 既時候，整斷R26下邊Q2連GND 既connection，令電流可以經R26 流到 (1) instead of GND。咁樣會即刻令電流由(3)去(2) SET左個CLK2，用果一下set 既raising edge 黎到令D flip flop 既output sync DATA2。好彩既係GP17係一個 10秒先轉一次既CLK（所以DATA2都係），有充裕既時間黎挫斷電線。我地無睇到果part 既code 以為 GP17 係~50 Hz 既 CLK 🤔 不過入到去見到咁慢就放心慢慢玩（有個LED D3 show 到個clk 係set 定 unset）

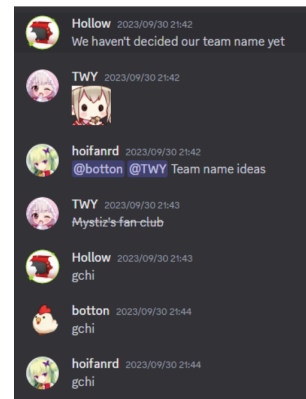


由於有時間限制同埋得一次機會所以都幾緊張，但最後都係時間內用個circuit diagram 同埋實圖寫好execution plan，要既cut 線既實際位置同方向，先後次序同埋唔同時間cut線既條件，最後順利 defuse 🤩 get flag。



SECCON Trip in Japan

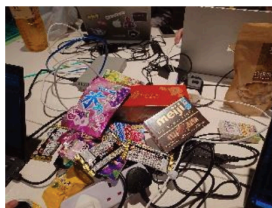
In December 2023, I joined the SECCON CTF Finals 2023 held in Tokyo, Japan, associated with the team Mystiz's Fan Club. Our CTF team actually has a great long history. I and one of my teammates, TWY, have been finding other CTF friends to gather a 4-man team every year since 2020, to join the annual HKCERT CTF (and we won almost every time!). This year, we have found the other 2 teammates, Hollow and botton, to join the HKCERT CTF 2023, and we have won the 3rd place in our category. Originally, only the 1st place will be invited to join the SECCON CTF Finals directly. However, since the 1st and the 2nd team abandoned the opportunity, therefore, we got the chance to join the competition.



This is how our team's name gets decided.



I am one of the participants of SECCON CTF Finals!



Are we coming for snack party instead?...

SECCON CTF is a well-known international CTF, and it is hard to get a position for its finals. We were excited to have a chance to join the competition. More essentially, the flight tickets were fully sponsored by HKCERT! Everyone knows that the flight tickets are extremely expensive during Christmas, and I am so glad that I got a free trip to Japan. Finally I went to Japan for 14 days, two days for CTF and twelve days for traveling :P.

Indeed we understood that we were the weakest among all the teams and were pretty sure that our team would be 1st on the scoreboard (counting from the bottom), but it was absolutely a great chance for us to learn (and to play). On the 1st day of the CTF, as a crypto player, I noticed that 5 crypto challenges were released, and I started to work on them. Unexpectedly, out of the 5 crypto challenges there were 3 challenges were about Quaternion (which I have no idea what that is). At that point I knew, I GGed. Anyway, I started working on *KEX 4.0*, which is DH Key Exchange on Quaternion, with TWY working on another crypto challenge (*DLP 4.0*) at the same time. A few hours later, TWY solved *DLP 4.0* while I was still working on *KEX 4.0*... (TWY so strong!).

I spent so many hours trying to solve *KEX 4.0*, I think I had spent like 2 hours finding the name of the scheme and another 3-4 hours on finding paper for the attack on that scheme. Finally I found a paper describing an attack on the scheme, I just implemented it and I got the flag, which I had totally no idea why it worked (Mystiz told me after the competition that he thought the attack himself at that time, too strong!). Afterwards, I started to work on *muck-a-mac*, which is a challenge about cracking ChaCha20_Poly1305.

The 1st day ended after I started working on that for some time. My teammates and I went back to the hotel for rest, planning to work hard on the next day --- and this is how we screwed up. Since all the challenges were released on the 1st day, all other teams continued to work on them during the night in the hotel, and every team (except us) submitted lots of flags immediately at the start of the 2nd day. Anyway, we didn't really care about the score a lot from the beginning... I tried to solve *muck-a-mac* for the entire day in Day 2, but I failed to solve it at last. I was in the right direction, but I was just bad at constructing lattices to perform the LLL...

In the end, our team was the last team on the scoreboard with 375 points (with the second last team having 1258 points...). However, I am still proud of myself since before the start of the competition, I said to myself that I would consider I have won if I could solve 1 crypto challenge, and I did it! This is a really precious experience to me --- this is the first time I go to Japan, this is also the first time for me to join an onsite CTF, which I enjoyed every moment that we worked as a team together, and I believed that it would be an unforgettable moment in my life.



Some places that I have been to during the Japan trip:

- UL – Maid Café
- (For Anime Pilgrimage)
- UR – Akihabara
- BL – Numazu
- BR – Comiket



異世界轉生黑客

Isekai Tensei Hakka

威噏變久噏？以 Black Bauhinia CTF Team 為題材的輕小說正式拖稿登場！如果你想 (or 唔想) 你出現在本小說中，請 Send 1BTC 畀 @ozetta，你的意見有可能被接納。

第一卷 — WOG FFA Battle

參考資料：

<https://github.com/blackb6a/blackb6a-ctf-2023-challenges/tree/main/20-isekai-tensei-hakka>

序章 — 虛擬即現實

"We are most likely live in a simulation — Elon Musk"

「哇，伊朗乜水又隊咗草？」塵夏幟說道。

突然在語音頻道出現了一句無關痛癢的話，令在場的隊友感覺有點錯愕。

這種感覺就像有人在頻道瘋狂點播韋琅硯的《來吧倒轉地球》，但又不是那麼糟糕。

我隨口說了一句：「夏幟，下次定再下一次交 Flag 係幾時？」

他回了一句：「█ you 祁屏梵」，我沒有再理回他。

雖然我心裡是這樣想的，但是我還是有點好奇那個伊朗乜水說了甚麼。

原來他說現實生活中的世界很可能是模擬的。

這也毫不意外吧，現實世界有這麼多不合理的現象，例如光速是常數。

正當我在發呆的時候，語音頻道突然傳出了 "First Blood" 的音效。

原來是鄧山貓他們解了 Crypto 題目。

塵夏幟又說：「█，乜依家 DeathCoin CTF 啲 Web 題重難過 Pwn 題」

我接著說：「係囉，連 Source Code 都唔畀，成堆通靈題玩條毛」

之後在語音頻道裡，鄧山貓的隊友路海帆點播了《唔好意思呀我要退出群組啊》……

然後沒有然後了。

隔了半日，語音頻道突然傳出「拍定手先 Yes」，Reverse 隊的李澤言正在直播。

果然他們有 Flag 了。還在食焗豬扒飯的我回覆了一個 Orz 的表情符號。

又隔了一日，連龍少年也拿到 Flag 了。看來 Web 題真的比 Pwn 題難。

「快啲通靈啦，再唔通靈食白果啦。」塵夏幟說道。

如果通靈是從另一個世界獲取資訊，那麼另一個世界又是甚麼？

正當我還在想怎樣通靈出原始碼時，我的眼前突然一黑……

原來我已經三日沒睡了……

第一章 — 首次轉生

當我回過神之後，我發現自己身在一個奇怪的空間。

「這是夢境嗎？」

「你當係啦(笑)」一陣謎之音從不知哪裡傳來。

眼前出現了一個浮動的藍色視窗，看來是老土的異世界轉生者系統呢。

我大聲叫：「システムコール、インスペクト、エンタエア、コマンドリスト」

一陣沉默過後……

「你是咪玩太多星爆棄療斬星光連流擊？」謎之音說道。

……專屬舞台裡，你可以進行動作選擇。

「好啦見你咁慘，你試下對住個窗向右摔。」

我向右摔了下，突然畫面竟然變成了 Sublime Text，周圍環境也不知為何變得模糊。

我心諗：「好彩唔係 vim 啫，如果唔係都唔知點撇返出去。」

「vim 好失禮你咩」謎之音突然響起。

「蛙跳～」我驚訝的叫了一下。

……

「拿，咪話我唔醒你，呢個叫 Za Warudo no Sosukodo」

「咩 Sudoku 話？」我帶有一點不滿地說著。

「Source Code 啊，頭先唔係君日本語本当上手架咩，依家成碌木咁。」

「點解無 flag.txt？」我問了謎之音。

「你唔係連死咗都要玩 CTF 下話」

原來我死了？

「講笑啫，無 flag 啊。現在是享受異世界的時間，慢慢玩啦。猜 Harder……」

我還想問 Dockerfile 在哪裡，但是謎之音沒有理我了。

我向左摔回那個藍色視窗，看到『創造新角色』的按鈕。我點了一下。

首先系統叫我輸入玩家帳號。還註明『★請勿輸入{ }; < > , " ' \ 等符號<』

最後那個小於是甚麼意思？手民之誤？隨後我向右摔了一下。

我在原始碼庫裡找了到 `wog_chara_make.php`，看來只是個表單生成的介面。

這個『創造新角色』的系統最後是通往 `wog_act.php`，其代碼像水蛇鬚咁長。

因為在 `wog_chara_make.php` 的第60至61行有兩個隱藏的表單參數，

(分別是 `f=chara` 和 `act=save`)，我找了 `wog_act.php` 第 77 行中相應的程序：

```
$wog_act_class->chara_save($bbs_id)
```

這是甚麼？原來 `chara_save` 是在 `class/wog_act_chara.php` 裡面。

裡面有一大堆檢查條件呢……有一個還檢查能力值是否負數，誰會填負數啊？

我再想了一下，好像還真的有個異世界小說的主角把顏值填負數來增加其他能力。

最有趣還是這個檢查條件：

```
if(((int)$ _POST["str"]+(int)$ _POST["smart"]+(int)$ _POST["agl"]  
+(int)$ _POST["life"]) > $wog_arry["total_point"]){//error}
```

看到這我感覺到有點問題。但是我心諗不是人肉 php interpreter 啫。

「試下向左摔啦」謎之音突然回應著。

我就順勢向左摔了兩下。結果出現了 `chrome://newtab` 的畫面。

這古怪的系統竟然有瀏覽器！

角色作成



異世界轉生黑客

Isekai Tensei Hakka

威噏變久噏？以 Black Bauhinia CTF Team 為題材的輕小說正式拖稿登場！如果你想 (or 唔想) 你出現在本小說中，請 Send 1BTC 畀 @ozetta，你的意見有可能被接納。

第一卷 — WOG FFA Battle

參考資料：

<https://github.com/blackb6a/blackb6a-ctf-2023-challenges/tree/main/20-isekai-tensei-hakka>

第二章 — 神奇的類型轉換

正當我看著那個奇怪的瀏覽器畫面蠢蠢欲動時，我決定向右摔兩下再看看原始碼。我對著 `class/wog_act_chara.php` 的原始碼，碌落啲去到 108 至 118 行，見到一些類似 `($_POST["str"]+8)` 的東西，看來是加了 8 點的基礎能力值呢。(註：`str` 是指 `strength` 而不是 `string`。`int` 是指 `integer` 而不是 `intelligence`) 但是剛才的檢查條件不是有甚麼 `(int)` 嗎？為什麼這裡又不用呢？難道 `(int)$x+8` 和 `$x+8` 有分別嗎？舊版本的 PHP 好像還真的有呢。我向左摔了兩下，然後在網址列搜尋了和 PHP 版本更新相關的資料，結果發現：

```
1 k?php
2 $x = "1e308"
3 var_dump((int), intval() ... integer operators and other conversions
4 var_dump(now always respect scientific notation in numeric strings.
5 var_dump
6 var_dump
```

看來舊版本的 PHP 不太尊重科學記數法。(韋琅硯謎之音：「請尊重 PHP」)

隨後我到 3v4l.org 驗證了一下：<https://3v4l.org/GnuCN>

果然在 PHP 7.1 之前，`(int) "1e308"` 會變成 1，但是 `"1e308"+8` 還是很大。

Output Perform 當我發現這個結果後，我急不及待摔回系統畫面來試試創造新角色。

帳號就叫 ozetta 吧，反正也不能輸入奇怪的符號。密碼？當然是 P@ssw0rd。

Output for 這符合了無用的密碼政策 — 「至少八個字元，含大小英文字、數字和符號」

之後要填玩家 email，這異世界還能收 email？又要摔去詭異的瀏覽器？

Output for 咁多要求，今次試下向下摔啦」謎之音又摔不及防地出現。

我向下摔了一下，竟然看到一個類似操作系統的桌面，還有個 email 的 App 圖示。

看來系統是平板電腦！難怪話 CTF player must know how to play CTF with Tablet.

Output for 「有無 SuperSU？」我輕輕一問。「你話呢？錯呀」

我向上摔回主系統畫面。反正從原始碼看來，這個異世界登錄也不用電郵認證。

除了電郵還有首頁名稱和位置，這些亂填就算了。至於角色性別.....還是填男好了。

Output for 一會突然在現實變性的話好像很危險。(順帶一提我 是男性。)

「你填女都無分別架，個名變粉紅色啫。」謎之音突然向我吐嘈。
「不了，紅字為笑點。我不想成為笑點。」我無奈地回應。
突然系統畫面上浮出了一個⓪的圖案。看來我又被吐嘈了。
之後還有角色屬性，「毒啦，你咁毒」謎之音又調戲我。我也順了牠的意選了毒。
「牠？你是咪未死過？咦你好似真係死過」……謎之音出現之後又突然陷入了沉默。
……至於出生地點，選第一個中央大陸就好了。之後要選角色圖像，還可以預覽呢。
當我打開時，裡面的圖竟然全都是叉包。「將就下啦，你揀咩都係用返你依家個樣」
那我就選 1 號角色算了。初始職業有戰士、術士和盜賊。「揀術士啦，大賢者」
「你唔畀我係滲透術士」我還是選了術士。「滲透？滲漏你就有啦」我沒有理會它。
下面是重點 — 角色能力！很不幸我只能選 +1 至 +10，我不知點算好，不知點算好，
「心中只感燥暴？向上摔啦」我隨後向上摔，竟然出現了瀏覽器的開發人員工具。
我心想這到底是甚麼鬼系統，還是我現在參與了甚麼整人節目。
我熟練地把四個能力選單中的 `<option value="1">` 的 1 改成 1e308
為甚麼是 1e308 而不是 2e308 或者 1e309？首先 $1+1+1+1 = 4 < 10$ 符合條件。
其次是浮點數最大值 $2^{**}1024$ 大約是 $1.79769e+308$ ，再大會變成 INF 會出錯。
然後我就急不及待地按了角色作成，按完才發現還有個必殺技名稱沒有填……
隨後我就發現周圍的景色不同了。「歡迎來到中央大陸」另一個謎之音說道。

我看了一眼自己的屬性值，力量、智力、敏捷、生命都是 65536，可惜魅力、信仰、體質還是 8。最後物攻、魔攻、物防、魔防也是 65536。看來一出山就能輾壓全場。
我再看了一眼周圍的景色，心諗「點解劍與魔法世界有現代汽車？」
「你問 Stable Diffusion 啦，唔關我事。」謎之音回答「你唔係都有全息投影電腦咩」
原來那個所謂系統是全息投影電腦？真是高科技。
我先到附近的修行地方試試自己的實力。我去到了中央大陸的中央平原挑戰。
這次先練練物理攻擊吧。物理系法師最高～（註：這裡的物理系不是大學主修物理科）
我遇到了一隻……毒蠍？看到那外觀真是不禁吐嘈。
「徵怪物圖片 bbs.2233.idv.tw」原來毒蠍長這樣啊。「將就下啦，哪有錢起異世界」
……看來這裡的經驗值不太夠。我之後到了中央大陸的水晶之間。這次試試魔攻吧。
遇到了一個叫「戰爭水晶」的怪物。是 EVA 的第五使徒嗎？
「徵怪物圖片 bbs.2233.idv.tw」很不幸，我心裡想像的八面體水晶結果長成這鳥樣。
結果毫無懸念地被我打了一下就 gg 了。獲得經驗值 410028475！獲得金錢 2041。
等級上升！力量上升 110、速度上升 106、智力上升 494、生命上升 207、體質上升 112、魅力上升 111、信仰上升 229。65536 不是最高嗎？怎麼還能上升呢。
我再看看角色狀態，果然屬性值還是 65536，但是 HP 上限從 45 升到 5251267，等級也從 1 級升到 201 級呢。隨手一揮就升了 200 級，看來異世界系小說沒有騙人。
為了補充 HP，我去了住宿。結果系統彈出了對話窗「金額不足 需6030元」
為什麼不是旅店的前台服務員說，而是系統用這樣的方式跟我說呢？
「將就下啦，哪有錢起異世界」謎之音又彈出了這句。
隨後我又去水晶之間掃蕩一下。這次等級又提升了，但是系統竟然跟我說力量上升 0
看來只升一級的效益不太好呢。這副本還要等 15 秒才能挑戰下一個怪物。真無語。
闖蕩了一分鐘之後終於有錢去住宿了。系統提示「休息了一晚後，HP回復精神飽滿!!」
感覺不錯。現在我的 HP 是 5277481/5277481，應該不會又死了吧。「又？」
我隨便在中央大陸逛逛，看到有個世界冠軍的告示板。現在的世界冠軍是叫 test，
性別女 215 歲，等級 1，HP 45/45。215 歲還只有等級 1 和 HP 45 是甚麼玩法……

Credits and Afterwords

- Editor-in-chief: *GonJK*
- Article contributors:

<i>a1668k</i>	<i>apple</i>	<i>botton</i>	<i>cire meat pop</i>
<i>Eason</i>	<i>ensy</i>	<i>GonJK</i>	<i>grhkm</i>
<i>harrier</i>	<i>hoifanrd</i>	<i>holloω</i>	<i>Jimmy</i>
<i>Mystiz</i>	<i>Ozetta</i>	<i>Stdor</i>	<i>streamline</i>
<i>vikychoi</i>	<i>vow</i>	<i>ωωkenωong</i>	
- Design: *apple*
- Cover art: Generated with Stable Diffusion by *apple*. Firebird Chan character by Starry Miracle.
- Article review (Knowledge-wise):

<i>apple</i>	<i>cire meat pop</i>	<i>grhkm</i>	<i>hoifanrd</i>
<i>Ozetta</i>	<i>ωωkenωong</i>		

If you have any comments on the newsletter, please don't hesitate to drop a direct message through Facebook, X (Formerly known as Twitter), E-mail, or even Discord – let us know what's on your mind!

As you dive into the articles, I hope you feel the passion and dedication that went into each piece. Thank you once again to our incredible writers and reviewers. Your contributions are deeply valued, and hope you enjoyed these articles.

Connect Us



blackb6a



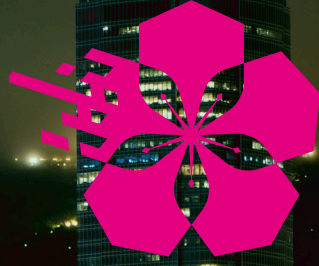
blackb6a



team@b6a.black



blackb6a



Black Bauhinia

<https://b6a.black>