



# Bauhinia Newsletter

Vol. 03 • September 2025

Vol. 03

## Contents

- Making 20 HKD Bad USB
- Writing Neovim Plugin with Rust
- CTF Cryptography Guidance
- Cloud Computing Features on Cloudflare
- DLL Hijacking
- Windows Malware Development
- Geometry Dash CTF Challenges
- Competitive Programming Tricks
- Third-party Authentication Bypass
- ...and more*





# Table of Contents

|                                                                                                         |    |
|---------------------------------------------------------------------------------------------------------|----|
| Table of Contents                                                                                       | 1  |
| Foreword                                                                                                | 2  |
| Ad-hoc                                                                                                  |    |
| Puzzle Solution – <i>Mystiz</i>                                                                         | 3  |
| Isekai Tensei Hakka Vol 1 Issue 5 – <i>Ozetta</i>                                                       | 4  |
| Isekai Tensei Hakka Vol 1 Issue 6 – <i>Ozetta</i>                                                       | 6  |
| Knowledge                                                                                               |    |
| 窮人 Bad USB – <i>GonJK</i>                                                                               | 8  |
| 小妹學用Rust寫Neovim Plugin小記1 – <i>grhkm</i>                                                                | 10 |
| CTF Crypto(graph) Guidance (Chapter 1) – <i>Eason</i>                                                   | 13 |
| CTF Crypto(graph) Guidance (Chapter 2: Common Stuffs about Cipher) – <i>Eason</i>                       | 15 |
| Introduction to Cloud Computing Features on Cloudflare – <i>Starry Miracle</i>                          | 16 |
| APT Technique Studying - DLL Hijacking – <i>botton</i>                                                  | 18 |
| Windows Malware Develop (?) series - (1) Understanding PE header & Create a new section – <i>a1668k</i> | 22 |
| Creating Geometry Dash CTF Challenges #1 - The Basics – <i>ivanwong13768</i>                            | 26 |
| Speeding up Different Types of Convolutions – <i>happypotato</i>                                        | 30 |
| 第三方應用設定失誤實例 – <i>vikychoi</i>                                                                           | 34 |
| Events                                                                                                  |    |
| How to plagiarize challenges for HKCERT CTF – <i>Mystiz</i>                                             | 36 |
| Co-Learning Project Trial Week Evaluation - <i>ensy</i>                                                 | 38 |
| Credits and Afterwords                                                                                  | 40 |

# *Foreword*

Welcome to the third 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 workaround the security measures 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.](#)



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

Participated in CTF events

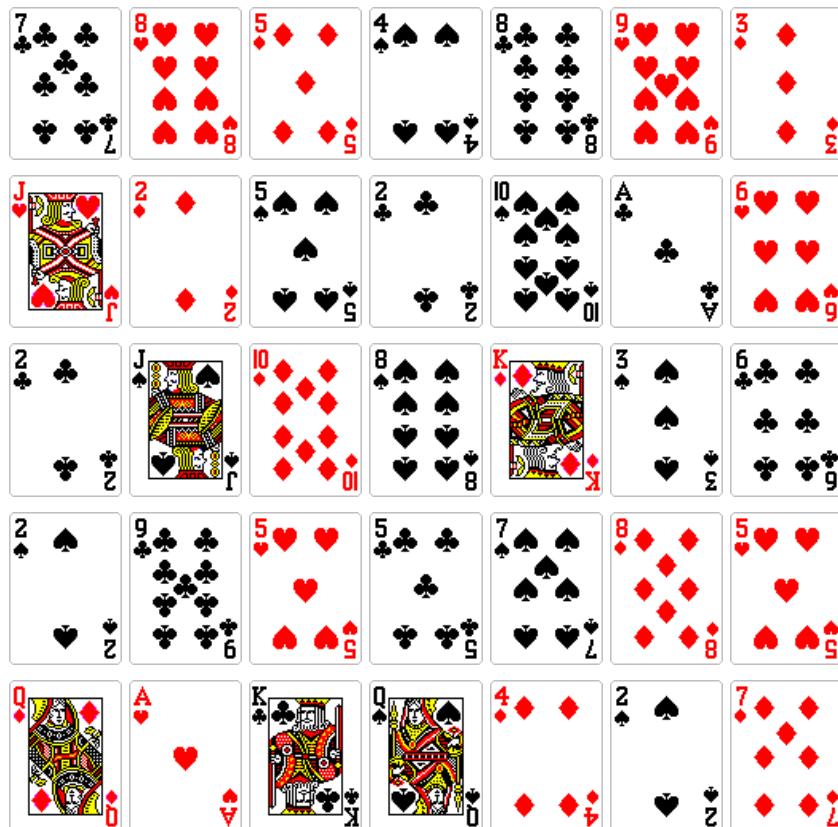
2025   2024   2023   2022   2021   2020   2019

Overall rating place: 54 with 417.670 pts in 2024  
Country place: 1

Looking forward, we are committed to continuing our efforts in organizing, participating in 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!



What were the cards facing down? 



Last time, we only revealed 12 cards. The cards arrangement was generated by the two patterns:

- Clubs and diamonds will only be shown on the odd positions, while spades and hearts will only be shown on the even positions.
- When we correspond A to 01, B to 02, ..., Z to 26, and map 01 to 13 to red cards and 14 to 26 to black cards, it spells “*the quick brown fox jumps over the lazy dog*”.

Congratulations to *hoifanrd* who successfully solved the challenge and sent me a hand of five cards: R2C2, R2C4, R3C1, R4C1 and R5C6. The cards formed *five of a kind*, twos. However, it is possible to form *five of a kind*, fives.



# 異世界轉生黑客

# Isekai Tensei Hakka

威喻變久喻？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 罢 @ozetta，你的意見有可能被接納。

## 第一卷 — WOG FFA Battle

**付費即可下載無刪減版**

參考資料：

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

## 第五章 — 裝備投影

只見系統提示「任務完成」，然後就沒有其他很有用的資訊。我問系統：「無啦？」謎之音回答：「無啦，你重想點，啲錢都畀你係咁變」，隔了一回，謎之音又說：「不如你自己變個任務獎勵畀你自己」，隨後系統彈出了另一個主線任務。  
「任務二：裝備一個不存在的裝備」我看到這詭異的任務後一臉愕然。  
我心想反正完成任務也沒有獎勵，我就放置不管它吧。「限時三日」謎之音突然說。  
我問：「三日之後會點？」謎之音回答：「任務失敗囉」「咁任務失敗會點？」  
「無架，可能你會永遠醒唔返掛」謎之音譏笑著說道。「呢度好食好住醒唔返咪好」原來我好像還未死。

結果我忽視了那狗屁任務。到了傍晚，我嘗試在這個離譜的異世界找一找音驚機舖。我走到街尾的公廁，轉角就看到有個娛樂場所。我走上了三樓，竟然真是音驚機舖。最神奇的是這裡竟然出現了某 Fan Club 的隊友們。hollow 和 hoifanrd 正在玩洗衣機，而 botton 和 TWY 正在一旁食花生。我也走到一旁食花生。「咦 ozetta？一齊玩哩」正在摔洗衣機門的 hoifanrd 一邊說著。我再看看自己的裝備，心諗無洗衣機手套點玩.....後來我又想起那狗屁任務，看來要自己變一對手套了。我仔細看了一會原始碼，發現 `INSERT INTO `wog_df`` 指令句只出現在 `wog3_sql_utf8.sql`。這任務真的能完成嗎？再看看裝備是怎麼顯示：在 `class/wog_act_chara.php` 之中第 221 行的 `show_chara` 函數，裝備名稱是用一堆 `left join` 來顯示的。這看起來沒甚麼注入的位置。那不如用客戶端攻擊？相信大家都知道惡名昭彰的 `document.designmode='on'`，「你咁改都唔算完全任務啊」謎之音補充。那麼如果我能在裝備顯示頁自動加料再自動加一件裝備呢？「算啦，咁算啦」.....我再看看哪裡能注入代碼。`p_url` 和 `p_homename` 好像不錯，但短得可憐。

看看如何砌.....

在 `wog.js` 裡，角色的公眾視點是以 `status_view2` 來顯示。裡面相關的內容：

```
'<a href="'+p_url+'" target="_blank">'+p_homename+'</a>'
```

另外在 `wog_act_chara.php` 中第 274 行是以單引號把這兩項內容傳到客戶端。那麼要怎樣利用只有 100 字的 `p_url` 和 30 字的 `p_homename`？如果 `p_url` 是 "`><svg/onload='$p1/*`, 那麼在 `p_homename` 就可以用 `*/$p2'`，從而增加字數。那麼 `$p1` 就能用 82 字元, `$p2` 就能用 25 字元。看起來能做到很多事。不過我這次想試試注入 CSS, 感覺比較直接。我先用開發人員工具的 Copy Path, 找出了手部裝備欄位是 `/html/body/table[1]/tbody/tr[9]/td[2]`。所以是第 9 列第 2 欄。對應的代碼是 "`><style>tr:nth-child(9)`

```
td:nth-child(2)::before{content:"洗衣機手套"}</style><a , 及後,  
0,p_url=0x223e3c7374796c653e74723a6e74682d6368696c642839292074  
643a6e74682d6368696c642832293a3a6265666f72657b636f6e74656e743a  
22e6b497e8a1a3e6a99fe6898be5a597227d3c2f7374796c653e3c61'295
```

在 PK 設定提交並重新登入後, 我的裝備欄多了洗衣機手套！終於可以玩洗衣機了？

「任務完成」系統竟然 [REDACTED] 有反應。當我開始摔洗衣機時, 感覺手有點不舒服。

假的裝備果然沒有效果。於是 I 決定弄個真的手套, 就 id=312 的「大賢者手套」吧。於是我在 PK 設定弄了 0,d\_hand\_id=312, 確認完再登入, 看到我又多了一對手套。「洗衣機手套大賢者手套」？感覺怪怪的。如何把大賢者手套的字眼隱藏掉？

此問題留待讀者解答。(提示: 試下好似上面個 XSS Payload 呀拆開)

我把大賢者手套隱藏起來後, 洗衣機的畫面已經出現了一堆 Miss 了。「za ga」

正當 hollow 串串貢地聊著, 我偷看了一下他的畫面, 發現他也是一堆 Miss,

雖然他的 Miss 數沒我那麼多。這場比賽結果是 hoifanrd 獲勝。「我去個廁所先」hoifanrd 正向著地下的公廁走下, 「咁我去買枝雪糕先」hollow 接著說。

我也去看看自動售賣機, 竟然有賣白果味雪糕.....然後我發現售賣機旁也有洗手間。

我正在思考為什麼現在的情節跟上次 **塵夏幟** 去日本通宵打機都唔玩 CTF 呀似果陣, hollow 突然轉身然後撞倒了我, 他手上的雪糕隨即飛濺在地上。[REDACTED]

我還在發呆地想著 [REDACTED] 雪糕是咁呢喺味還是白果味時, hollow 隨口一說：

[REDACTED]  
「頭先成碌木咁企響我後面 jm9, Sai-Sai 唶雪糕啦。」我還是躺在地上發呆著。

謎之音：「你肯定你唔係為咁擺甫士？敏捷 65535 的主角」我無言以對。

「做咩無晒反應？」[REDACTED].....系統突然彈出了一大堆「[REDACTED] 警告」

[REDACTED] 為了系統不要發神經又 Skip 了這段美好回憶,

我決定增加一點其他的美好回憶。我隨後到 PK 設定輸入了：

```
0,p_url=0x223e3c7374796c653e74643a3a61667465727b636f6e74656e74  
3a222b [REDACTED] 227d3c2f7374796c653e3c61 WHERE  
p_name=0x686F6C6C6F77 -- .
```

我按下確定後, 我看到 hollow 全身突然多出了一堆裝備投影。是甚麼就留給讀者猜。

「今次你 [REDACTED]」我笑著說。「[REDACTED] 警告解除」.....系統無奈地回應。

「[REDACTED] 好核突啊, 快啲幫我整走佢」此時 hoifanrd 也回到了三樓,

「哇你 [REDACTED]」.....系統又突然彈出一堆「[REDACTED] 警告」.....

算了吧, 讓它 Skip 吧。我幫 hollow 的 `p_url` 設定成 `null`, 妨礙他的裝備也消失了

[REDACTED]  
「我敢講出完呢段古之後, 以後響 Discord Channel 無人敢講『[REDACTED]』？」



# 異世界轉生黑客

# Isekai Tensei Hakka

威喻變久喻？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 紐 @ozetta，你的意見有可能被接納。

## 第一卷 — WOG FFA Battle

付費即可下載無刪減版

參考資料：

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

## 第六章 — 富康街寵物公園

在異世界中住唐樓養番狗是甚麼體驗？首先你要有狗。所以我去到商店街的牧場，然後一個對話框「沒有寵物資料」出現，令我十分失望。怎樣才能拿到新寵物呢？

我向右捽了系統畫面一下，然後尋找 `insert into wog_pet`，看來有兩個方法。

首先有個在 `class/wog_mission_tool.php` 中 `mission_pet_get` 的函數，這個函數很詭異的只有在 `mission/wog_mission_73.php` 用過。看看描述：

「完成條件：捕捉 幻獸 路行鳥」「任務獎勵：寵物 幻獸 不死鳥」路行鳥換不死鳥？這時我

另一個獲得新寵物的方法是從 `class/wog_fight_select.php` 的第 136 行：

`if($p[d_name] == "捕捉器" && !$pet ) .....` 簡單來說是要先裝備捕捉器，

然後還要符合隨機條件 `if(rand(1, 25) == 1) .....` 真麻煩，煩過捉寵物小精靈。

我隨後去到商店街的道具屋買了一個捕捉器，「今次唔變啦咩？」謎之音疑惑地說。

「738 蚊啫」我冷冷地回覆它。買完後我就去裝備它，然後在原始碼找找哪裡有狗。

「地獄惡犬.....」看起來將會是一條乖巧的狗，「在 `m_place=4`，是迷霧森林吧」

我從中央大陸出發，穿過了獅子山後來到了熱帶雨林中的迷霧森林。「獅子山？」

謎之音疑惑地問，我沒有理它。打了幾十回合後，那破爛的捕捉器還是沒有發動。

果然穿過了非洲，運氣也變得非洲？然後我發現身上的捕捉器竟然消失了！[bbs.2233.idv.tw](http://bbs.2233.idv.tw)

「捕捉器損壞」算了，我又買了一堆捕捉器再試試。「捕捉器損壞」「捕捉器損壞」

「ching 你是咪呃稿費？」謎之音問我，我答「是」。謎之音無語。我也很無奈，

異世界的捕捉器衰過小米。又打了幾十回合後，終於遇到了地獄惡犬，我隨手一揮：

「ozetta 發動 連續攻擊 327hit 紿予 地獄惡犬 4.59245671465E+12 點傷害」

毫無懸念變成死狗了，可是捕捉器又損壞了。我都開始懷疑是不是我出手太重？

「4% 應該易過抽 SSR 呢，你個隨機數係咪壞架」我不滿地吐嘈。「Try Harder~」

OZETTA 發動 連續攻擊 327hit 紿予 地獄惡犬 291222E+12 點傷害

「ozetta 發動 連續攻擊 5818hit 紿予 地獄惡犬 126906996420 點傷害」

這次 hit 數多了但是攻擊力少了.....「捕捉器損壞」咁落去寫到呢章完都未有寵物。

終於在幾百回合後，捕捉器有反應了。「捕捉到 機器人一號」，算了，先留著吧。  
我走到商店街的牧場，看看這寵物的數據。「AT, MT, DEF 都是 1.....」果然是一號。  
「你的隨機數是不是壞了」我正經地問，謎之音以呂 [REDACTED] 的口吻回答：「沒有唷～」  
然後系統畫面浮出了 :bcgugu2: 的表情符號。這東西還有 2 呢？究竟有甚麼意義.....  
我還是想養狗，就把寵物的名改成「IT狗」，結果寵物的名變成「IT狗-機器人一號」  
寵物的外觀也從人型的機器人變成狗型的機器人。我吐嘈：「依家科技咁先進架咩」  
「IT狗，你的資訊真的很有用」我對著寵物說，它就興高采烈地撲向我身上。  
但是防禦力只有 1 的 IT狗應該很快會爆肝死吧。所以我打算出手幫它大改一番。  
在原始碼裡尋找 `update wog_player` 時，找到 `class/wog_act_pet.php` 的第 165 行：  
`"update wog_pet set pe_st=".$_POST["temp_id"]." where pe_p_id=".$user_id." and pe_id=".$_POST["pay_id"]"`  
這個 `pet_chg_st` 函數看來是變更狀態吧。我再去牧場看看，狀態有攜帶和獸欄。  
這裡沒有提交表單鍵，但會觸發 `parent.foot_trun('pet','chg_st',1,0)`  
我把最後的參數 0 改成 '0,pe\_def=9e9'，注意這裡要把單引號加上來表示字串。  
點下選項後，「設定成功!!」再看寵物的狀態，DEF 65535，這樣 IT狗 變得堅固了。  
隨後我萌生了一個奇怪的想法，「如果 Dr. ROT10 變成我的寵物會怎樣？」  
謎之音緊張地回應：「杰哥不要.....」我心諗我又不是胡 [REDACTED] 然後刷了 :bcgugu2:  
(黃韋建：我想睇呂 [REDACTED] 大戰胡 [REDACTED]) 謎之音沉默了。我再把寵物的狀態從 0 改成：  
`0,pe_name=0x44722e20524f543130,pe_at=9e9,pe_mt=9e9,pe_fu=9e9,pe_hu=0,pe_he=9e9,pe_hi=9e9,pe_mname=[REDACTED]`  
更改狀態後，寵物名稱變成「Dr. ROT10-[REDACTED]」。我哈哈一笑，謎之音也震驚了。  
只見眼前的 IT狗 突然變成了一名白毛的獸耳少年，[REDACTED]。  
牧場裡的人和狗看到紅都面晒。「好耐無見喎 ozetta.....主人」Dr. ROT10 彆扭地說  
我得意地說「[REDACTED]」「主人，我本身是機器人一號，所以我是一號」  
原來寵物的怪獸編號還是最初的 38。但是這個世界只有機器人一號和機器人二號，  
我心生一計，然後久喻著：「世界上只有機器人一號和二號，你之前不是 IT狗 嗎？」  
「哪隻 IT狗 會用『Option Base 1』的邪魔外道？陣列首個索引值是零，所以你是零」  
Dr. ROT10 貌似被我昆了，「好的主人，我是零」他疑惑地回應著。全場一片愕然。  
我帶著 Dr. ROT10 離開了牧場。從以前的「親家」變成現在真的親家，感覺有點怪。  
我們再回到迷霧森林冒險，這次又遇到地獄惡犬。我隨手一揮，地獄惡犬又死了。  
一旁的 Dr. ROT10 好像在發呆一樣。看來能力太強的話，寵物也只是裝飾品。  
我整天都在大戰，Dr. ROT10 整天也沒有出擊過。「說好的出擊值 255 呢？」  
晚上回到酒店，前台服務員問：客官要不要把寵物寄放在獸欄？「不了，慳錢。」  
謎之音吐嘈：「[REDACTED] 我比較好奇 NPC 是怎樣知道他是寵物？」  
  
謎之音面露難色地問。隨後系統又出現了個主線任務：「任務一：殺死 Dr. ROT10」  
我一面疑惑，這個任務不是之前已經完成了？「你知任務失敗會點啦？今次限一日」  
「你永遠都無法叫醒一個裝睡的人」我蔑視地回應它。系統隨後彈出「任務變更」。  
沒有系統的打擾，看來以後就能過著 [REDACTED] 現充生活。  
Dr. ROT10 在房間裡找到了一支 King Blade 應援棒，不愧是他，出擊值都用到這裡。  
「讓我好好支援你吧主人」系統畫面隨後彈出 :blowcatglowsticks: 的表情符號。  
  
當然是打 CTF 啦！我把 `flag{Byte}` 交到系統後，拿到了 1 分。應援棒也閃耀著。



## 序

是咁的，由於要教啲中學生玩 Bad USB，所以我要物色一個好嘅 solution 比啲學生玩。

你會係篇文感受到滿滿嘅怨氣。

## What is BadUSB?

BadUSB 係泛指模擬 Human Input Interface (HID) 嘅設備以執行攻擊。係 2010 年嘅時候，Hak5 founder Darren Kitchen 就本身想利用 Keystroke Injection 去自動執行一啲單調又平凡嘅工作<sup>1</sup>。

由於 BadUSB 係模仿人類去打字，自己寫 payload 去 launch 攻擊。佢無直接一個 malware 嘅 payload 會放係電腦上面，增加 detection 難度。

係市面上商品化咗嘅 BadUSB 有 USB Rubber Ducky 握，而且佢哋有配合嘅 Programming Language 叫 DuckyScript。但係，作為一個價格敏感型消費者，要比接近 900 港幣去買係肉痛的。

[Home > Rubber Ducky](#)



RUBBER DUCKY  
€89,00  
World famous USB Keystroke Injection Device. What you could do in minutes, the Rubber Ducky does in milliseconds.  
Abuse the trust that computers give to keyboards - at over 1000 words per minute.

Quantity

- 1 +

ADD ACCESSORIES

- Rubber Ducky Pocket Guide €19,95
- Rubber Ducky Textbook €49,99

## 破

所以，我地有無得利用工業革命嘅力量，將價格壓下去呢？自己動手，豐衣足食，當年師傅教落可以用 Teensy 3.2 去做 BadUSB。但係？

The screenshot shows a product page for the "Teensy 3.2 ARM Cortex-M4 Development Board". It features a photograph of the board, which is a small rectangular PCB with a central microcontroller and various pins. The page includes product details such as the part number (NTS760), price (NT\$762), and shipping information. There are also social media sharing buttons and a "Buy It Now" button.

無啊！無貨啊！壞一隻少一隻啊！官方叫你 upgrade 上 Teensy 4.0 啊！上 Teensy 4.0 賣你 200 大洋啊！

The screenshot shows an eBay listing for the "Teensy 3.2" development board. It displays the product image, price (AU \$420.00), and a "Buy It Now" button. The listing is marked as "Brand New". Below the main listing, there are buttons for "Add to cart" and "Add to Watchlist". A note indicates that people are checking it out.

ebay 賣你 2000 港紙？我是盤子嗎？

## Life will find a way

其實我要利用 Teensy 嘅 HID class 同 Arduino IDE 去 simulate HID input。是否意味我可以更便宜嘅 Arduino Nano 去做 BadUSB?

思路正確，但係 Arduino Nano (ATmega328P) 原生無支援 HID<sup>2</sup>。所以我將目光投向 Digispark (ATtiny85) 同 Arduino Leonardo (ATmega32U4)，兩款都可以 native 支援 HID。

The screenshot shows a product page for the "ATTiny85 Micro Mini USB Development Board". It features a photograph of the small, rectangular microcontroller board with a USB port. The page includes product details like the part number (ATTINY85-04PU), price (¥7.3), and shipping information. There are also options for different colors and a "Buy It Now" button.

係某寶上面最平搵到 7.8 港紙嘅 Option。

<sup>1</sup> <https://docs.hak5.org/hak5-usb-rubber-duky>

<sup>2</sup> [https://www.reddit.com/r/arduino/comments/saq6zl/atmega328p\\_as\\_usb\\_keyboard\\_controller/](https://www.reddit.com/r/arduino/comments/saq6zl/atmega328p_as_usb_keyboard_controller/)

# 急

買咗少少 Digispark 返嚟開工做 test，即刻發現：

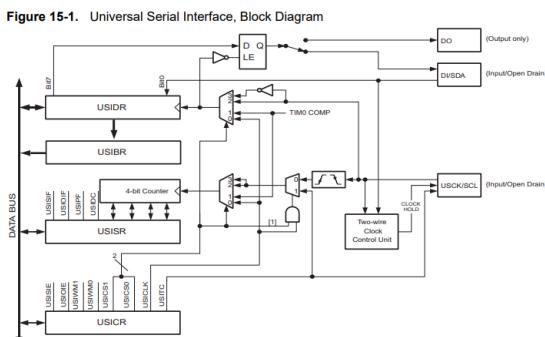
- Digispark 插落電腦無反應！

第一時間我諗嘅係，我係咪又浪費咗錢錢？Google 一輪發現好似係插 USB 3.0 嘅 port 就會出事，連 Device Manager 同 Zadig 都無新野。關於我點知道可以 downgrade USB 2.0 解決請睇 this<sup>3</sup>。

## V-USB 迷思

尋根究底想知道我衰咩，發現咗有人係 Reddit 話 DigiSpark 用 V-USB 係軟件層面模擬 HID，無硬件上嘅支援。USB 3.0 controller 要求精確嘅時脈但 Digispark 模擬唔到，you shall not pass。<sup>4</sup>

DigiSpark 係結構圖<sup>5</sup> 上面無 SPI (Serial Peripheral Interface)，實際上使用 USI (Universal Peripheral Interface) 模擬 SPI 同 I<sup>2</sup>C  
◦ 情況同 Bit Banging 利用 software 同 GPIO 模擬 Serial Communication 有類似地方。



一圖解釋 ATtiny 85 USI 點模擬 SPI

有人講過試下改 Bootloader 會 work，no luck。

<sup>3</sup> <https://www.youtube.com/watch?v=MmDBvgrYGZs>

<sup>4</sup>

[https://www.reddit.com/r/arduino/comments/w5f77m/digispark\\_attiny85\\_on\\_usb\\_3/](https://www.reddit.com/r/arduino/comments/w5f77m/digispark_attiny85_on_usb_3/)

<sup>5</sup> <https://www.mouser.com/datasheet/2/36/doc2586-70764.pdf?srsltid=AfmBOOrL-tmmMiVoAamOo3jv1a12YvTVMe7S35WB19NX3fRSfYkZdFdc>

## The Show Must Go On

咁我唯有買多幾條 USB 2.0 延長線返嚟做 test。

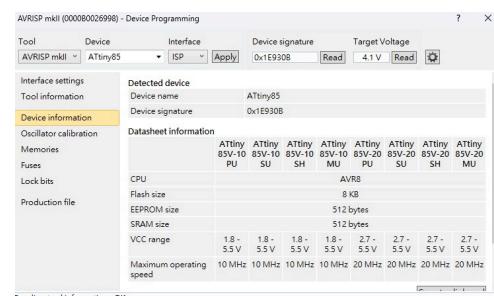


結果佢條線 Bad Contact 哒咗我兩隻 DigiSpark。  
一條 5 蚊嘅線整咗我兩隻 9 蚊嘅 Digispark，我嘅心情 exactly 同下圖一樣。



## 終

係朋友幫助之下，搵咗隻 USBasp 吉針試下燒返個 Bootloader 入 Digispark 又行得走得。原來只係無咗個 Bootloader，有如霍金再世唔洗輪椅。在此特別感謝蕉哥 (@banana8964) 幫我燒錄。



結論：用 ATmega32U4 可以排除 Driver 同 USB 問題，launch attack 唔要求 victim 有裝 Driver。雖然唔知點解我買嘅 Arduino Leonardo 平過 ATmega32U4，20 蚊之內有交易。



## 1 前言

是噏的，小妹作為老人家，做得多 ctf 自然見得多題目。大部份 target 「有手有腳」嘅人就做到嘅 ctf 都會有勁多簡單 crypto 題，例如以下呢類題目（簡化咗）：

```
p=309934309611001980924396626832706072993
q=203352468583722278782880952545761971181
e=65537
ct=19110958343511268718924875534205741716588748547599782292763995798235404060198
```

當然，以我嘅 pattern recognition 能力同打字速度嚟 solve 呢啲題目完全唔係問題，只不過呢啲題目做得多都悶㗎嘛，於是我就決定寫 code 勿 automate 呢類題目。我原本諗住寫個 Discord bot command，但係我實在唔想再掂我嗰七百幾行 TypeScript code，所以都係寫返啲 local script 算。小妹用開 Neovim，所以最後決定寫段 Rust 再 bridge 去 Neovim 度。

首先講咗 Crypto 嘅 part 嘅 Rust 先啦，眾所周知就「其實唔難」，最煩嗰 part 係搞掂個 borrow checker 同條 Rust 到寫 bigint，我就揀咗用 `num_bigint` 入面嘅 `BigUint` 寫所有嘢。Decrypt 嘅段 code 大概係噏：

```
// src/rsa.rs
use num_bigint::BigUint;
use num_traits::One;

#[derive(Default, Debug)]
pub struct PrivateKey {
    e: Option<BigUint>,
    p: Option<BigUint>,
    q: Option<BigUint>,
}

impl PrivateKey {
    // ...
    pub fn decrypt(&mut self, ct: BigUint) -> Result<Vec<u8>, String> {
        if let (Some(e), Some(p), Some(q)) = (&self.e, &self.p, &self.q) {
            let n = p * q;
            let phi = &n - (p + q) + BigUint::one();
            let d = e
                .modinv(&phi)
                .ok_or_else(|| "Failed to compute modular inverse")?;
            Ok(ct.modpow(&d, &n).to_bytes_be())
        } else {
            Err("One or more of e, p and q are not provided".to_string())
        }
    }
}
```

接住落嚟我地要將 Rust 同 Neovim 嘅 API 駁埋一齊。寫呢啲 plugin 最緊要就係睇 docs（如果唔想慢慢睇，TJ DeVries 嘸 YouTube upload 咁段九個半鐘嘅 audiobook，應該幫到你）。我哋好容易就搵到 Neovim 入面有 `jobstart` 呢個 command，which 據我理解就係 Neovim 勿 spawn 個 process + handle 呢啲 serialisation/IO。喺 Rust 嘅邊我就揀咗用 `nvim-rs` 呢個 crate。佢前身係 `neovim-lib`，我原本亦都係用嗰個 crate，但係去到後面發現有啲 API 無 interface，再發現原來個 crate 晨早 archive 咁，所以最後要成個 plugin 寫過。

## 2 同 Rust 打乒乓波

喺寫 RSA 嘅 part 之前，等我哋一齊嚟睇吓點樣用 `jobstart` 同 `nvim-rs` 啦！古語有舊云，呢部份其實唔難嘅，我哋首先寫個好 minimal 嘅 Neovim config file 去 load 個 binary：

```
-- config.lua
local bin = "./target/release/nvim-rsa-plugin"
local rsJobId = vim.fn.jobstart(bin, { rpc = true })

vim.api.nvim_create_user_command("Ping", function(opts)
    print(vim.rpcrequest(rsJobId, "ping"))
end, { nargs = 0 })
```

見到我哋用 `jobstart` 去 run 個 binary，攞返個 job id 返嚟，再將 `:Ping` 用 `rpcrequest` 駁去個 binary 度。值得注意嘅係我用咗 `rpc = true` 呢個選項。其實唔用 RPC protocol，但係你就要人手 encode/decode data，用 `chansend` 同 `stdin/stdout` 嘚做 IO，which 係煩好多。另一樣嘅係呢度用 `rpcnotify` 其實都 ok，但係陣間 Rust 嘅邊要 somehow 彈返個 result 去 Neovim 嘅邊再 display 出嚟，所以會煩少少。

之後我哋就可以寫 Rust 嘅部份啦。其實呢度唔少都係抄 `nvim-rs` 嘅 examples，所以大家可以參考吓。另外我有用 GitHub 嘅 search 功能嚟睇吓其他用 `nvim-rs` 嘅 plugin 係點運作。Anyways 我拆開咗兩個 file，第一個係個 entry point：

```
// src/main.rs
pub mod eventhandler;
pub mod rsa;
use nvim_rs::create::tokio as create;

#[tokio::main]
async fn main() {
    let handler = eventhandler::NeovimHandler {};
    let (nvim, io_handler) = create::new_parent(handler).await.unwrap();
    // TODO: Better handling
    let _ = io_handler.await.unwrap();
}
```

見到段 code 有咩特別，純粹整個 event handler 再開個 event loop。第二個係個 event handler 嘅 implementation：

```
// src/eventhandler.rs
use async_trait::async_trait;
use nvim_rs::{compat::tokio::Compat, Handler, Neovim, Value};
use tokio::fs::File as TokioFile;

#[derive(Clone)]
pub struct NeovimHandler {}

#[async_trait]
impl Handler for NeovimHandler {
    type Writer = Compat<TokioFile>;

    async fn handle_request(
        &self,
        name: String,
        args: Vec<Value>,
        nvim: Neovim<Self::Writer>,
    ) -> Result<Value, Value> {
        match name.as_ref() {
            "ping" => Ok("pong".into()),
            _ => unimplemented!(),
        }
    }
}
```

呢度段 code 大部份都係 boilerplate，個重點係 `handle_request` 呢個 function，個 `name` 係個 request 嘅名(即係 `rpcrequest` 嘅第二個 argument)，而 `args` 顧名思義係 Neovim 嘅邊嘅 arguments(例如 `:CommandName arg-1 arg-2 arg-3`)

三個 args)。最後嘅 nvim 係而家 active 嘅 Neovim session，下面我哋寫 RSA 個部份就會見到點用佢㗎喇。

而家我哋只要 run 下面呢三行 command：

```
$ cargo build --release
$ rm -rf /
$ nvim -u config.lua
```

再打 :Ping，就會見到個 pong 嘅 response 哟喇！

### 3 RSA decrypter

而家我哋睇吓點樣寫 RSA 嘅部份啦，只要將上面 handle\_ping 嘅 code 改做下面呢個 handle\_rsa（再 cast 返個 error）就得㗎喇。我哋直接睇 code：

```
// src/eventhandler.rs
impl NeovimHandler {
    async fn handle_rsa(
        &self,
        nvim: Neovim<Compat<TokioFile>>
    ) -> Result<Value, Box<CallError>> {
        // obtain current buffer in neovim
        let buf = nvim.get_current_buf().await.unwrap();
        let content = buf.get_lines(0, -1, false).await.unwrap();

        // load file content into a private key object
        let mut sk = PrivateKey::new();
        let mut ct: Option<BigUint> = None;

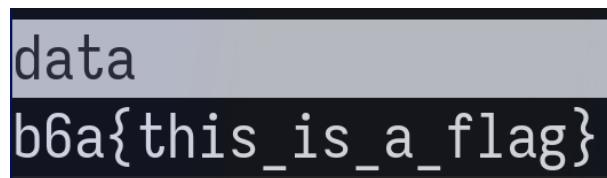
        for line in content {
            if let Some(resp) = sk.load_line(&line) {
                ct = Some(resp);
            }
        }
        let ct = ct.ok_or_else(|| Value::from("ct is not provided"))?;

        // decrypt ciphertext
        let dec = sk.decrypt(ct)?;
        let dec_string = String::from_utf8_lossy(&dec);

        // display virtual text
        // ...

        Ok(dec_string.into())
    }
}
```

上面嘅 sk.load\_line 就留返畀讀者自己寫啦，只係 Regex 同 set 返回 parameter。同埋見到我漏空咗 display virtual text 嘅 implementation，呢個就留返下次先再寫。而家我哋只需要將開頭嗰四行 RSA 寫落個 file 度，再 run nvim -u config.lua <filename> 同 :RSA（擺返你改嘅 command 名），應該就會喺 status bar 度見到下面噃樣：



```
data
b6a{this_is_a_flag}
```



## Um, what is this?

Welcome to the first chapter of my series, "CTF Crypto(graph) Guidance." In this series, I aim to share valuable insights and practical tricks for tackling cryptography challenges in CTF, drawn from my own experiences. Although the difficulty won't strictly increase with each chapter number, you can expect the content to grow more technical and zero in on particular aspects of cryptography as the series progresses.

As this is the first chapter, it's worth starting with an introduction to the mindset and habits that can save you significant time when tackling crypto challenges, regardless of their difficulty.

## Will it make me cry?

Well, it depends on how much you hate math, but I hope it never gonna make you cry.

## What should I prepare?

Maths sense, Programming skill, strong logical thinking, etc. But passion is the most important prerequisite as it is the only motivation to learn more about cryptography topics.

## 1 Everything is Maths

For most of the easy to intermediate level (maybe later define the level lol) crypto challenges, you would be given source program file and its output (usually flag as input but encrypted) or the interactive platform (usually a TCP server or web interface).

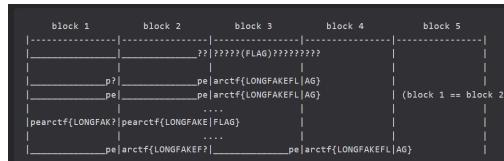


Figure 1: Note for AES-EBC encryption oracle while doing PearlCTF 2024 crypto/security++

doing crypto challenge.

However, a large part of these is breaking widely used encryption schemes which are improperly implemented. The math may seem complex, but more often than not, a simple understanding of the underlying principles will allow you to find flaws and crack the code. Drawing graphs and equations is an effective way to understand the implementation. It is always a good practice to take notes on iPad or even paper while doing crypto challenge.

## 2 Try to make use of everything

In CTF challenges, we would try to find the bug with provided "clues". In cryptography, we would have some known & unknown variables and equations which is the clues. We can either attack the protocol itself, which are usually difficult, or attack implementations of such protocols (e.g. bad random number generation, bad prime number generation, key reuse, etc.). Therefore, Figuring out what we have given is very important. For example, we might be given 2 equations and some outputs which represent 2 different encryption process but with same key elements and input. Then we can crack the input by the relation of the 2 equations (e.g. Broadcast Attack in RSA, Franklin-Reiter related message attack, etc.).

### 3 If you have no clue, stop it, get some help.

It's easy to feel lost in a crypto challenge, staring at a wall of numbers and code with no idea where to start. Don't panic—it happens to everyone. The key is knowing when to pause and seek help. The internet is your friend here: a quick search on the cryptography topic at hand can unlock a better understanding of the concepts or reveal similar problems others have solved. Whether it's RSA, AES, or some obscure cipher, there's likely a blog post, paper, or forum thread that sheds light on it. Beyond that, don't hesitate to lean on third-party tools and libraries. Need to whip up a solve script? Libraries like PyCrypto, SageMath, or even simple Python modules can handle the heavy lifting—say, [RsaCtfTool](#)—so you can focus on the logic. The goal isn't to reinvent the wheel; it's to crack the flag. So, if you're stuck, step back, Google it, and let the tools do some of the work.

### 4 If you get stuck, you are inside the rabbit hole

We've all been there: hours deep into a challenge, scribbling equations, chasing a solution that's just out of reach. That's the rabbit hole, and it's a time thief. When you're stuck, don't keep banging your head against complex math—pause and reassess. Ask yourself: Are my current methods even feasible with my skills? How much time have I sunk into this approach? Good time management is critical in CTF, so regularly evaluate if you're on the right track or just digging deeper into a dead end. Instead of fixating on the equations you've built, shift gears: try coding something up. Experiment with the outputs, inputs, or whatever you've been given. Look for patterns or quirks—can you combine equations to cancel out an unknown variable? Maybe there's a simpler path you've overlooked, like a flawed implementation rather than a mathematical breakthrough. Step out of the hole, rethink your strategy, and explore alternatives.

### 5 Bruteforcing is straight forward but sometimes brilliant

Bruteforcing gets a bad rap as the “lazy” option, but in CTF crypto, it can be a stroke of genius. Computers are fast—let them do the grunt work. If you're dealing with an unknown variable and its possible range is small enough (say, a short key or a limited character set), bruteforcing might just hand you the flag on a platter. For example, a weak random number generator or a low-bit encryption key could crumble under a few thousand guesses. That said, it's not always mindless trial-and-error—you need some math sense and CTF experience to spot when it's viable. Check the challenge setup: if it's a server with a limited number of attempts (like 100 guesses before it locks you out), that could be a subtle hint bruteforcing is the way to go. Pair it with a bit of analysis—narrow the range with what you know—and it's less “brute” and more “brilliant.” Just don't overdo it; know the limits of your tools and the challenge.

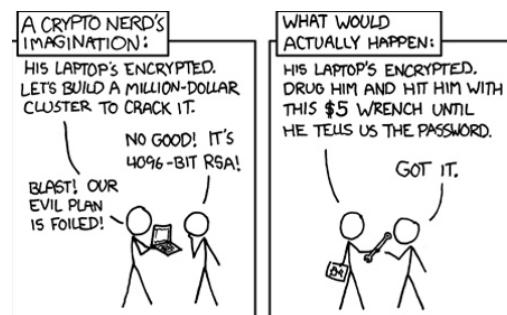


Figure 2: Bruteforce smarter



In this chapter, we would focus on common stuffs on Stream Cipher and Block Cipher.

## o Observation with Encoding

(Item o because it is not specifically about Cipher lol)

If you find something weird about the output value or text. Try to convert it to another encoding to observe the pattern. Let's assume there is a crypto challenge that gives the value of key xor flag, and its base64 value is `//+/33/v/9/`. This seems weird, right? When we base64 decode it and then convert to binary, we can find that there are many ones and nearly no zeros.

According to what we know about Exclusive or, in a bit, the xor result is 1 if and only if the inputs differ, therefore, the values of key and flag have nearly opposite bits.

## 1 About Stream Cipher

Stream Ciphers encrypt pseudorandom sequences with bits of plaintext in order to generate ciphertext, usually with XOR. However, because it uses XOR, so that means it might have vulnerabilities such as key reusing, especially when the challenge gives you two or more ciphertexts using the same key. We also need to keep in mind of some types of stream ciphers, e.g. [Sals20](#), and pay attention to the implementation like if the challenge missing the mixing rounds, which is an important step to make encryption invertible.

## 2 About Block Cipher

For most of the block ciphers, they would have mode of operation, such as AES, DES, SM4, etc. Each mode has their vulnerabilities and it is actually quite easy to find out the attacks. Such as ECB mode + allow encrypt infinitely attempts + plaintext is combined by input and Flag = Bruteforce character one by one with chosen plaintext (related challenge: [Cryptohack - ECB oracle](#)). CBC mode + allow encrypt infinitely attempts = padding oracles. Also, for bruteforcing, try decrypt one block only to save computation time, especially ECB mode.

## 3 About DES and AES

Besides mode of operation, the two most popular ciphers - DES and AES, also have interesting attacks according to their implementation.

**DES:** Most of them are (semi-)weak keys and improper key usage in [Python](#), which is due to its 56-bit key length. In Python, the required key is 64 bits long, but the effective key length is 56 bits only. So we can use a different key string but get the same effect on encryption.

**AES:** Modified AES round function is also one of the typical CTF topics. Which is playing around the 4 operations (AddRoundKey, SubBytes, ShiftRows and MixColumns). As AES is a Substitution-permutation network, so that means there are linear steps (e.g. MixColumns) and non-linear steps (SubBytes). The challenge [HKCERT CTF 2021 - Sratslla SEA](#) is a good example to understand each of them and why AES has such design. And it also requires a good sense of math and bruteforcing.



# Introduction to Cloud Computing Features on Cloudflare

## Starry Miracle

Disclaimer: “Not sponsored” by Cloudflare

做唔到Oracle免費仔就嚟Cloudflare “薅羊毛” 啦！(無用小知識：薅羊毛 is listed as a pentest item in CN)

With Cloudflare Worker, you can setup **Private AI**, **Discord Bot**, **(cron) web crawler**, and more...

The screenshot shows the Cloudflare Account Home interface. On the left is a sidebar with various services: Discover, Domain Registration, Log Explorer, Analytics & Logs, Security Center, Trace, WAF, Turnstile, Load Balancing, Interconnects, Magic Transit, Magic Firewall, L3/4 DDoS, IP Addresses, Zero Trust, Compute (Workers), Secrets Store (New), Workers for Platforms, Storage & Databases, R2 Object Storage, AI, Stream, Images, Realtime, Tag Management, Manage Account, Notifications, and Bulk Redirects. The main area is titled "Workers & Pages" under the "Compute" section. It lists three worker projects: "4m-galaxy-south", "4p-sun-east", and "4m-sun-south". Each project shows metrics like Requests, Errors, Median CPU Time, and Median Wall Time. A "View bindings" link is present for each. The top right of the main area has a "Create" button and a "Workers documentation" link. The top right corner also shows account details: "Live on June 24 at 11 am CDT", "Requests today: 372 / 100,000", "Observability events today: 1,348 / 200,000", and "Jun 1, 2023 - Jun 19, 2023". Below these are sections for "Account details" (free plan), "Manage API tokens", and "Subdomain: starrymiracle.workers.dev".

**Computer (Workers): Example of what you can see in Worker page.**

To create and develop a Worker project, Cloudflare recommends you use Wrangler. I am too dumb to understand how to use it, so I just edit code and configurations in the web interface (Visual Studio Code for the Web).

```

11  export default {
12
13    async scheduled(event, env, ctx) {
14      //TODO: update env variable
15
16      ctx.waitUntil(scheduledTask(env.Galaxy4pSouth, env.pending, env.sid));
17      console.log("cron processed");
18    }
19
20    async fetch(request, env, ctx) {
21      /*
22        if (request.method !== 'GET') {
23          return new Response('Method not allowed', {
24            status: 405
25          });
26        }
27
28        try {
29          //TODO: update env variable
30          return scheduledTask(env.Galaxy4pSouth, env.pending, env.sid);
31        } catch (err) {
32          return new Response(
33            JSON.stringify({
34              error: err.message
35            })
36          , {
37            status: 500,
38            headers: {
39              'Content-Type': 'application/json'
40            }
41          }
42        )
43      */
44      ctx.waitUntil(sleep(1000));
45    }
  
```

^ This is the sidebar, displaying the key functions Cloudflare provided. It is normal to have more/less features, depending on your contract/paid tier.

scheduled() – **Cron Triggers** will call this function;

fetch() – HTTP request will trigger this function;

queue() – consume request from **Queues**

You can also do subrequest within, e.g.

```

async function postToDiscord(content) {
  const discordWebhookUrl = "https://discord.com/api/webhooks/13";
  const discordResponse = await fetch(discordWebhookUrl, {
    method: "POST",
    headers: {
      "Content-Type": "application/json"
    },
    body: content,
  });

  if (discordResponse.ok) {
    return new Response(
      JSON.stringify({
        status: "Message sent successfully!"
      })
    );
  } else {
    return new Response(
      JSON.stringify({
        status: 500,
        headers: {
          "Content-Type": "application/json"
        }
      })
    );
  }
}
  
```

Other Related Features & Free Tier Limit: (May change from time to time)

| Name                    | Description                                                                                                               | Free Tier Limit                                                                                                                                |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Workers                 | (See previous page)                                                                                                       | <b>100,000 Req/Day</b><br>10 ms CPU time per invocation<br><i>*/workers/platform/limits/#subrequests</i><br><i>*/workers/platform/pricing/</i> |
| Subrequest              | Requests generated within one call to worker<br>(Fetch request, Access KV/D1 Database)                                    | 50 subrequests per request                                                                                                                     |
| Durable Objects         | Special kind of Workers that comes with storage.<br>(I have not tried it before as it is newly on free tier;<br>2025 Apr) | Only DO with SQLite storage backend,<br>no KV storage backend<br><b>100,000 Req/Day</b><br><i>*/durable-objects/platform/pricing/</i>          |
| Cron Triggers (Workers) | Cron job for Workers (smallest unit: 1 minute)                                                                            | Maximum 5 <b>Cron Triggers</b> per account<br><i>*/workers/platform/limits/</i>                                                                |
| KV                      | Key-value pairs that can be accessed by Workers.<br>Can be accessed by REST API/Cloudflare SDKs                           | Read: 100,000/day; Write/Delete:<br>1,000/day<br>Storage: 1GB<br><i>*/kv/platform/pricing/</i>                                                 |
| D1 Database             | Serverless SQLite databases                                                                                               | Rows Read: 5 million/day<br>Rows Written: 100,000/day<br>Storage: 5 GB (total)<br><i>*/d1/platform/pricing/</i>                                |
| Hyperdrive              | (Connector to your existing DBs, I have not tried before)                                                                 | 100,000 DB queries / day<br><i>*/hyperdrive/platform/pricing/</i>                                                                              |
| Queues                  | (Currently only available on Paid Tier, not tried before)                                                                 | N/A                                                                                                                                            |
| R2 Object Storage       | S3-compatible object storage. I have not tried before.                                                                    | Free tier is available, but it requires you to register payment method first.<br><i>*/r2/pricing/</i>                                          |

\* [https://developers.cloudflare.com/.....](https://developers.cloudflare.com/)

The limit would be more relaxed if you subscribe to the \$5 per month Workers Paid. Additional cost may be incurred if you consumed more than the limit available in Free Tier. (The free daily/monthly quota for requests/storage/etc. will still be there)

Also, it seems that there are more and more stuff that is available in free tier. You can try to run some mini projects on it without concerning the cost. However, you need to consider the number of web requests per day/month, so you may need to do some tricks to combine the functions/batch your SQL queries before sending out.



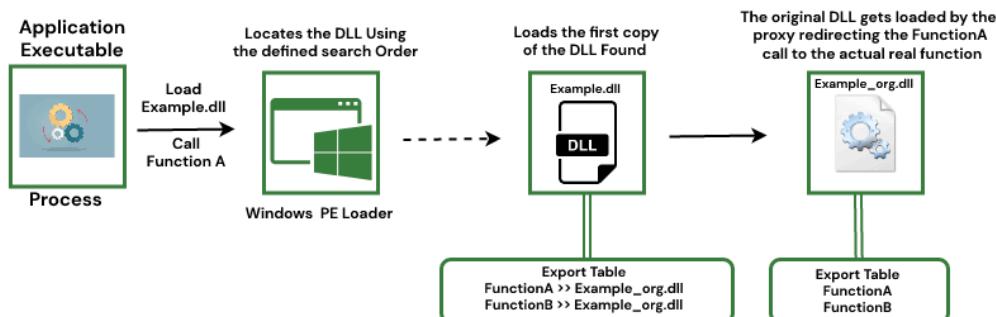
# APT Technique Studying - DLL Hijacking

button

**DLL hijacking** is a technique used to manipulate the execution flow of a legitimate application by abusing dynamic-link library (DLL) files. The attacker weaponize this technique to achieve **persistence, privilege escalation, and defense evasion**.

In this article, we will skip the fundamental explanation of what a DLL is and how it works. Instead, we will focus on what attackers aim to achieve with DLL hijacking, particularly in the context of Advanced Persistent Threats (APT).

## DLL Hijacking/Proxying



## Persistence

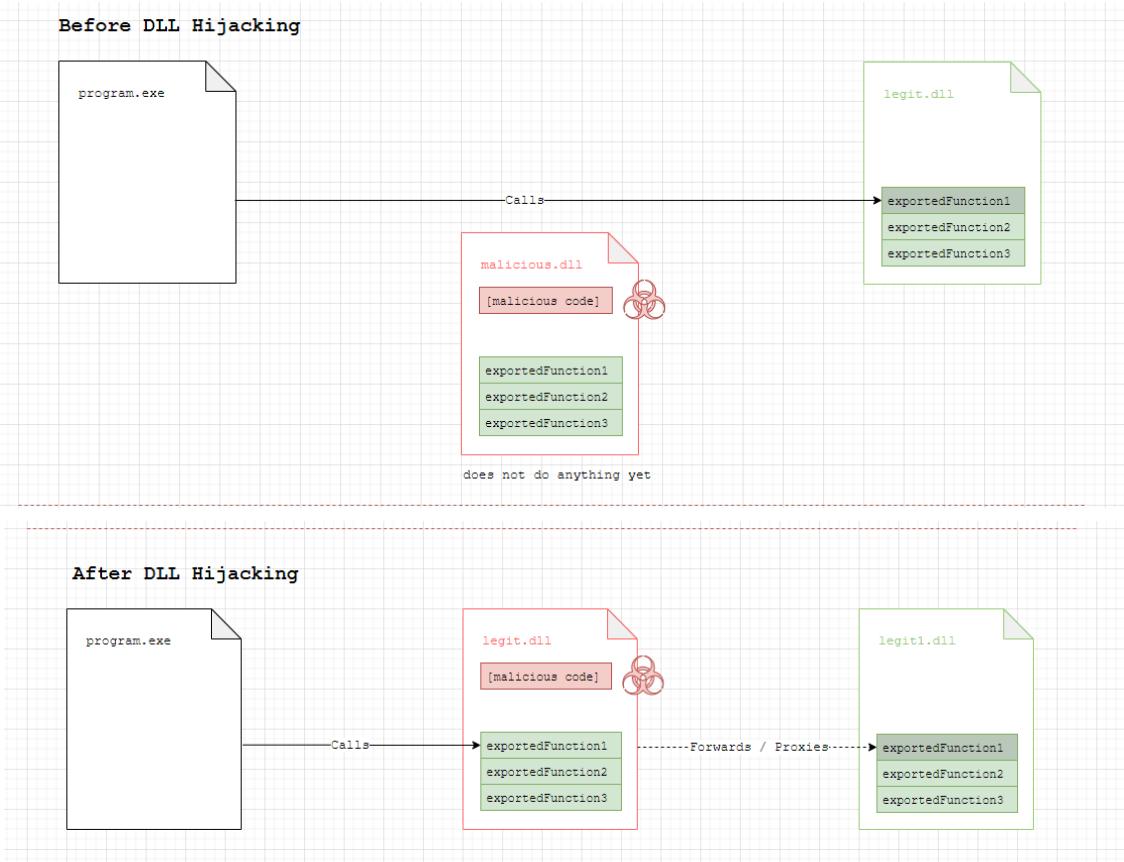
Persistence is a critical objective for attackers once they compromise a system. Traditional methods—such as scheduled tasks, registry modifications (e.g.,

`HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`), and service path alterations—are commonly used. However, these techniques are increasingly monitored and detected by modern antivirus (AV) and endpoint detection and response (EDR) solutions.

**DLL hijacking** offers a stealthier alternative for persistence. The attacker exploits the way Windows searches for DLL files by either:

- Replacing an existing DLL with a malicious one in a writable directory.
- Dropping a malicious DLL to fulfill a missing dependency.

When the targeted application runs and attempts to load the DLL, it unknowingly executes the attacker's code.



## Real-World Example: Microsoft Teams

An excellent example of this technique is covered in [MiloSilo's tutorial on Microsoft Teams Proxy DLL Hijacking<sup>1</sup>](#). According to the research:

- Microsoft Teams attempts to load certain DLLs from `%USERPROFILE%\AppData\Local\Microsoft\Teams\current`.
- If a required DLL, such as 'USP10.dll', is missing in that directory, the application still attempts to load it from there due to the DLL search order in Windows.
- By placing a **malicious proxy DLL** named `USP10.dll` into that folder, an attacker can ensure their code runs when the user launches Microsoft Teams.
- The malicious DLL can also forward legitimate calls to the original DLL, maintaining the application's normal functionality while executing the payload in the background.

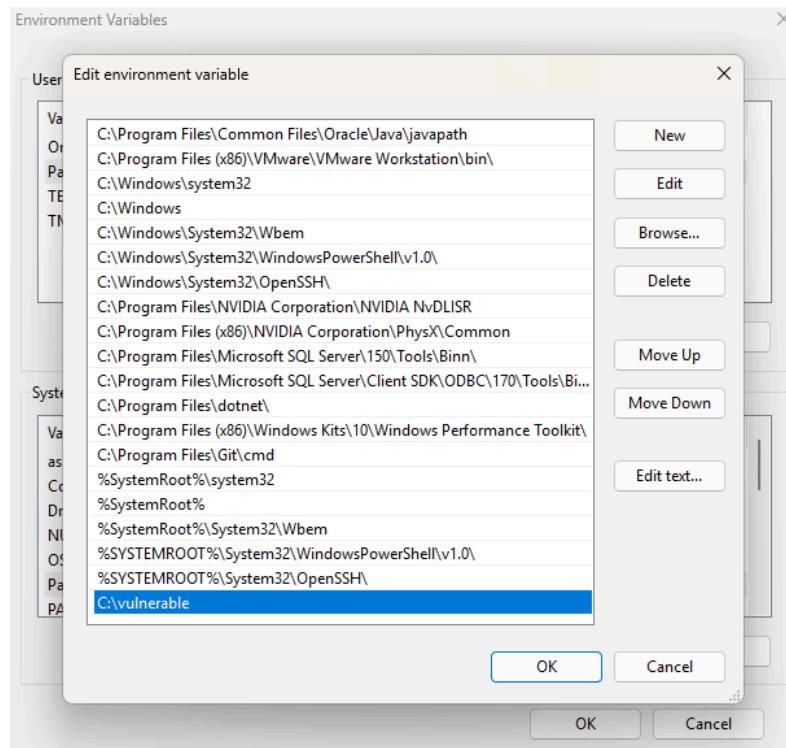
This technique allows the attacker to **maintain persistence** without triggering obvious alarms, as it exploits a trusted application and uses a legitimate DLL-loading mechanism.

## Privilege Escalation

DLL hijacking can also be leveraged to perform privilege escalation. One common scenario involves abusing writable directories found in the system's environment `PATH` variable.

<sup>1</sup> <https://milosilo.com/hacking/microsoft-teams-proxy-dll-hijacking/>

If a non-privileged user has write access to any directory listed in the system `PATH`, it becomes a potential vector for privilege escalation. This is because Windows services or system processes might load DLLs from these paths, often with elevated privileges, such as `SYSTEM`.



A practical example of this technique is demonstrated in the [WptsExtensions.dll](#)<sup>2</sup> project.

## Example: Task Scheduler DLL Hijacking

The `WptsExtensions.dll` technique targets the Task Scheduler service:

1. The attacker identifies a writable folder that exists in the `PATH` environment variable of the machine scope.
2. A malicious DLL named `WptsExtensions.dll` is placed in that writable directory.
3. A batch script (`script.bat`) containing the payload or further instructions is also added.
4. Upon system reboot, the Task Scheduler service auto-loads `WptsExtensions.dll` from the `PATH`, executing the malicious DLL with `SYSTEM` privileges.

This allows the attacker to escalate privileges from a regular user to `SYSTEM`, gaining full control over the machine.

To check the environment path, we can use `set` in cmd or

`[System.Environment]::GetEnvironmentVariable("PATH", "Machine")` in powershell to identify vulnerable paths and misconfigurations on the machine.

---

<sup>2</sup> <https://github.com/phackt/wptsextensions.dll>

## Evasion

In addition to persistence and privilege escalation, DLL hijacking can be an effective technique for evasion, helping attackers bypass security controls such as antivirus (AV), endpoint detection and response (EDR), and application whitelisting.

Choose to hijack legitimate and trusted applications that are already allowed or whitelisted in the environment. By injecting malicious code into a trusted binary through a hijacked DLL, the malware can piggyback on the reputation and trust of the legitimate application, making detection significantly more difficult.

In one of my red team exercises, I observed that the use of `WptsExtensions.dll` not only enabled privilege escalation, but also contributed to bypassing application control mechanisms.

| Processes                    |        |        |            |         |            |
|------------------------------|--------|--------|------------|---------|------------|
| Name                         | Status | 5% CPU | 37% Memory | 0% Disk | 0% Network |
| Service Host: Task Scheduler |        | 0%     | 5.5 MB     | 0 MB/s  | 0 Mbps     |
| Task Scheduler               |        |        |            |         |            |

This occurred because of the boot order of Windows services:

- The Task Scheduler service initializes early in the boot process.
- Many application control solutions (such as AppLocker, Carbon Black, or third-party endpoint control software) are themselves implemented as services that start later in the boot sequence.

As a result, by placing a malicious `WptsExtensions.dll` in a writable directory listed in the system `PATH`, the payload executes before the application control software is fully active. That early execution window allows the malicious DLL to run unrestricted—effectively bypassing application control policies that would normally block unsigned or unauthorized binaries.

## Conclusion

DLL hijacking is a stealthy and effective technique used by attackers to achieve persistence, privilege escalation, and evasion. By exploiting DLL search order and misconfigured permissions, malicious code can execute under trusted applications or system services. Understanding these techniques is essential for both offensive and defensive security operations.



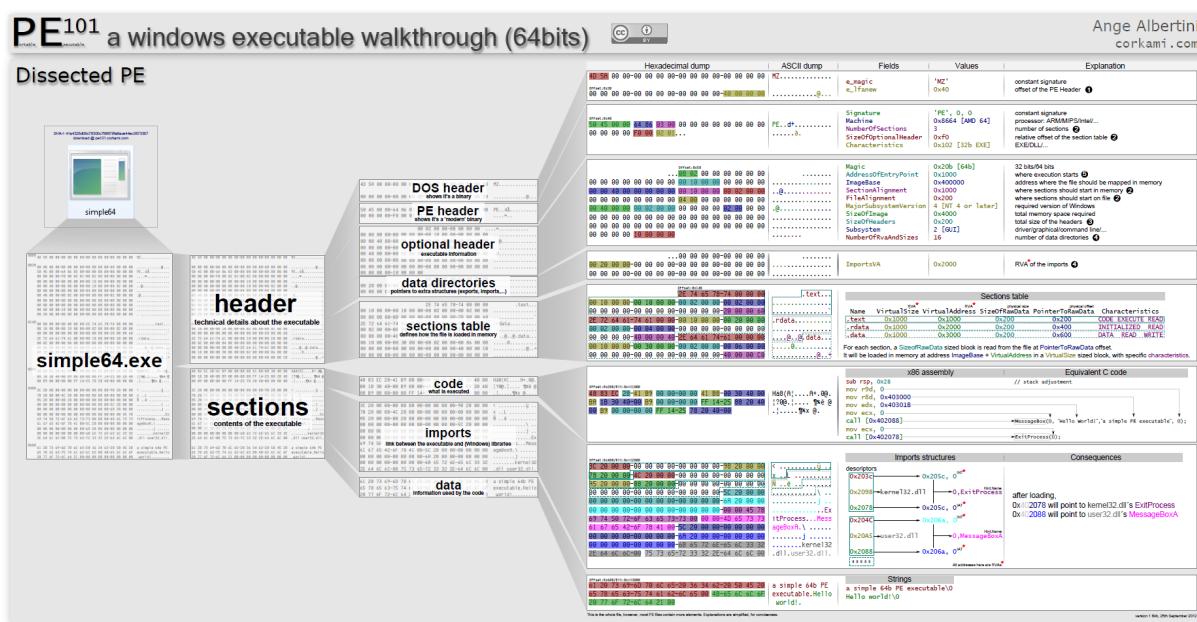
# Windows Malware Develop (?) series

## - (1) Understanding PE header & Create a new section

Recently, I joined a workshop about Windows and Linux Reverse Engineering instructed by Boris So in BSidesHK 2025. In that workshop, one of the exercises involved injecting a malicious shellcode inside legitimate software, Notepad++.exe. Because almost no one understands what Boris is talking about in the workshop (xDDD), and I am really interested in this topic, I would like to make a series about how to develop Windows malware (?) by injecting malicious code into some legitimate software. This series will be a tough series with lots of technical details, but I will try my best to explain all the steps in detail, from the structure of Windows executable, how to write a malicious shellcode, to maybe how to bypass different defenses and develop your first malware (?). Hope this journal won't get 腰斬 because I didn't have time to do research ;(. In this series, all the examples will be based on Notepad++ v8.7.8 (64-bit). If you want to follow my guide to build your own malware (?), feel free to download one from their official webpage.

As our ultimate goal is to inject codes into a Windows executable, we will need to understand how Windows Portable Executable (PE) works in order to be able to add a new code section to run our malicious codes. Therefore, the first chapter of this series will cover Windows PE Header (64-bit) in detail, and how to create a new code section. Let's get started :D

Before we go into the PE file structure, what really is a PE file? A Portable executable is a file format for executables used in Windows operating systems, which is based on the COFF (Common Object File Format). A lot of files, like executable files (.exe), dynamic link libraries (.dll), Kernel modules (.srv), and Control panel applications (.cpl) are PE files. A PE file follows the structure outlined in the following figure:



Reference: <https://github.com/corkami/pics/blob/master/binary/pe101/pe101-64.png>

Ok... Seems a little bit complicated... Let's break down each part.

(Notes: All the header offsets and values below are based on my notepad++ v8.7.8 (64-bit), you may need to find the offset and values of your own executable in case you are using another executable (or even if you install the same version of Notepad++); Also in case you like to read code to understand the details of PE Header, this C header file describe the structure of executable (image) and object files under the Windows family of operating systems: <https://github.com/jasonwhite/ducible/blob/master/src/pe/format.h>)

## DOS Header (Offset: 0x00000000 to 0x0000003F)

Every PE file starts with a 64-byte-long structure, which is called the DOS Header. This header contains only two important information, including:

- 1) [Offset: 0x00000000] **e\_magic**: The file signature (aka "magic bytes") – "4D 5A" ("MZ"), and
- 2) [Offset: 0x00000030] **e\_lfanew**: The offset of the PE Header. In my Notepad++ v8.7.8, the offset of the PE Header is 0x00000128 (remember Windows PE is in little-endian mode).

| Offset(h) | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F          | Decoded text |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------|--------------|
| 00000000  | 4D | 5A | 90 | 00 | 03 | 00 | 00 | 04 | 00 | 00 | 00 | FF | FF | 00 | 00 | MZ.....ÿÿ.. |              |
| 00000010  | B8 | 00 | 00 | 00 | 00 | 00 | 00 | 40 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .....@..... |              |
| 00000020  | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .....       |              |
| 00000030  | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 28 | 01 | 00 | 00 | .....(....) |              |

## DOS Stub (Offset: 0x00000040 to 0x000000127)

This is a tiny MS-DOX 2.0 compatible executable that prints out the message "This program cannot be run in DOS mode" when the program is not compatible with Windows. In our case, you don't really need to bother with this. But you can try to reverse this program if you want to xDD

| Offset(h) | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F       | Decoded text     |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------|------------------|
| 00000040  | 0E | 1F | BA | 0E | 00 | B4 | 09 | CD | 21 | B8 | 01 | 4C | CD | 21 | S4 | 68       | ...!..Li!Th      |
| 00000050  | 69 | 73 | 20 | 70 | 72 | 6F | 67 | 72 | 61 | 6D | 20 | 63 | 61 | 6E | 6F |          | is program canno |
| 00000060  | 74 | 20 | 62 | 65 | 20 | 72 | 75 | 6E | 20 | 69 | 6E | 20 | 44 | 4F | 53 | 20       | t be run in DOS  |
| 00000070  | 6D | 6F | 64 | 65 | 2E | 0D | 0D | 0A | 24 | 00 | 00 | 00 | 00 | 00 | 00 | 00       | mode....\$.....  |
| 00000080  | B3 | DE | 3F | 04 | F7 | B3 | 51 | 57 | F7 | B3 | 51 | 57 | F7 | B3 | 51 | 57       | >?.-?QW-?QW-?QW  |
| 00000090  | BC | CB | 52 | 56 | FD | B3 | 51 | 57 | BC | CB | 54 | 56 | 37 | B3 | 51 | 57       | 4ÉRVÀ?QW-ÉTV?QW  |
| 000000A0  | 94 | 30 | AC | 57 | F2 | B3 | 51 | 57 | 34 | 30 | 55 | 56 | E7 | B3 | 51 | 57       | 40-Wè?QW4OUV8?QW |
| 000000B0  | 34 | 30 | 52 | 56 | FB | B3 | 51 | 57 | 34 | 30 | 54 | 56 | E9 | B3 | 51 | 57       | 40RVG?QW4OTV8?QW |
| 000000C0  | BC | CB | 55 | 56 | E8 | B3 | 51 | 57 | BC | CB | 57 | 56 | F6 | B3 | 51 | 57       | 4ÉUVè?QW-ÉWV8?QW |
| 000000D0  | BC | CB | 50 | 56 | D0 | B3 | 51 | 57 | F7 | B3 | 50 | 57 | 98 | B1 | 51 | 57       | 4ÉPVD?QW-?PW-?QW |
| 000000E0  | E4 | 37 | 58 | 56 | C5 | B2 | 51 | 57 | E4 | 37 | 51 | 56 | F6 | B2 | 51 | 57       | à7XVÀ?QW?QV8?QW  |
| 000000F0  | E4 | 37 | AE | 57 | F6 | B2 | 51 | 57 | F7 | B2 | C6 | 57 | E7 | B2 | 51 | 57       | a7BWè?QW-?EW1?QW |
| 00000100  | E4 | 37 | 53 | 56 | F6 | B2 | 51 | 57 | 52 | 69 | 63 | 68 | F7 | B2 | 51 | 57       | a7SVè?QWRich-?QW |
| 00000110  | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00       | .....            |
| 00000120  | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 50 | 45 | 00 | 00 | 64 | 86 | 06 | 00 | PE..dt.. |                  |

## PE Header/NT Header (Offset: 0x00000128 to 0x0000022F)

This part is one of the most important parts of a PE file, which stores different kinds of information of this PE file. The main PE Header contains three different parts, 1) Signature, 2) File Header (aka COFF Header), and 3) Optional Header. We will need to modify these "a little bit" in order to be able to add a new code section. Time to dig deep into how each part works...

### 1) Signature (Offset: 0x00000128 to 0x0000012B)

Basically, is a fixed value of "50 45 00 00" ("PE\0\0") which represents the start of the PE Header. The offset also matches with value we discovered in the DOS Header (e\_lfanew).

### 2) File Header (Offset: 0x00000128 to 0x0000013F)

The File Header (aka COFF Header) consists of 20 bytes of a PE file and holds the information about the physical layout and properties of the PE file. There are a total of 7 members in this section, including:

- 1) [Offset: 0x0000012C] **Machine**: Tells you the information about the types of machine (CPU), in my case, value 0x8664 represents x64 machine.
- 2) [Offset: 0x0000012E] **NumberOfSections**: The number of sections in the Section Table (will cover more later).
- 3) [Offset: 0x00000130] **TimeDateStamp**: A Unix timestamp that indicates when the file was created.
- 4) [Offset: 0x00000134] **PointerToSymbolTable**: The offset of the entry point to the COFF symbol table.
- 5) [Offset: 0x00000138] **NumberOfSymbolTable**: The number of entries to the COFF symbol table.  
(Note: Both 4) and 5) will be set to 0 because COFF debugging information is deprecated.)
- 6) [Offset: 0x0000013C] **SizeOfOptionalHeader**: The size of the Optional Header (will cover more later).
- 7) [Offset: 0x0000013E] **Characteristics**: The flags that indicate attributes of the object or image file, if you want to know more about these flags, please check the official Microsoft documentation by yourself UwU: <https://learn.microsoft.com/en-us/windows/win32/debug/pe-format#characteristics>, the whole thing is too complicated if I need explain how it works here (and also not enough space for me to write it too)...

| Offset(h) | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E            | 0F            | Decoded text |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---------------|---------------|--------------|
| 00000120  | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 50 | 45 | 00 | 00 | 64 | 86 | 06 | 00            | .....PE..dt.. |              |
| 00000130  | C2 | 4B | CB | 67 | 00 | 00 | 00 | 00 | 00 | 00 | F0 | 00 | 22 | 00 | ÀKëg.....ë.." |               |              |

| Offset(h) | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E            | 0F    | Decoded text |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---------------|-------|--------------|
| 00000120  | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 50 | 45 | 00 | 00 | 64 | 86 | 06 | 00            | ..... |              |
| 00000130  | C2 | 4B | CB | 67 | 00 | 00 | 00 | 00 | 00 | 00 | F0 | 00 | 22 | 00 | ÀKëg.....ë.." |       |              |

To achieve our goal – adding a new code section, we will have to add 1 into the **NumberOfSections**.

Hence, the value of **NumberOfSections** will be 0x0007 now.

| Offset(h) | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E            | 0F            | Decoded text |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---------------|---------------|--------------|
| 00000120  | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 50 | 45 | 00 | 00 | 64 | 86 | 07 | 00            | .....PE..dt.. |              |
| 00000130  | C2 | 4B | CB | 67 | 00 | 00 | 00 | 00 | 00 | 00 | F0 | 00 | 22 | 00 | ÀKëg.....ë.." |               |              |

### 3) Optional Header (Offset: 0x00000140 to 0x0000022F)

Welcome to the start of the nightmare (?). Although this header is called "optional", it is **essential** for the execution of the **PE file**. The reason it is called optional is because some of the file types (e.g. dynamic link libraries (.dll)) don't contain this header. The reason why I think this part is a nightmare is simply because it contains 30+ fields. Also, there are two versions of the Optional Header: one for 32-bit (PE32) and one for 64-bit system (PE32+). Some of the field sizes are changed from 4 bytes into 8 bytes for the 64-bit version, and the 32-bit version contains an extra field, "*BaseOfData*", which is absent in the 64-bit version. As a reminder, the following examples are based on the 64-bit version (PE32+). Therefore, the offsets are only correct if you are reversing a 64-bit Windows executable.

Because for the sake of simplicity (and the length of this article), I won't cover all the fields. You can read <https://learn.microsoft.com/en-us/windows/win32/debug/pe-format#optional-header-image-only> if you want to know all the fields in the Optional Header. But I will cover some of the important fields for you:

- 1) [Offset: 0x00000140] **Magic**: A field that tells whether an image is 32-bit ("0B 01") or 64-bit ("0B 02").
- 2) [Offset: 0x00000150] **AddressOfEntryPoint**: The address where the Windows loader will begin execution.  
This normally holds the *Relative Virtual Address (RVA)* of the *Entry Point (EP)* of the module and is usually found in the *.text* section.
- 3) [Offset: 0x00000160] **SectionAlignment**: A value that is used for section alignment in memory. It must be  $\geq$  **FileAlignment**, and the default value is the page size for the architecture, which is 0x1000 in x86\_64.
- 4) [Offset: 0x00000164] **FileAlignment**: The alignment of sections in the file. The default value is 0x200.
- 5) [Offset: 0x00000178] **SizeOfImage**: The size of the image file, including all the headers and code sections.  
It has to be a multiple of the **SectionAlignment** value (0x1000 in our case). In our example, our current image size is 0x82E000.
- 6) [Offset: 0x00000180] **Checksum**: The checksum of the image file. One interesting fact is Windows **will not** check the checksum when you run an PE file, but only check it when you run a DLL or driver. Therefore, you don't really need to change it when you are modifying the header of a Windows executable.
- 7) [Offset: 0x00000184] **Subsystem**: Identifies the target subsystem for an executable file, i.e. which Windows subsystem is required to run the image. In our case, 0x2 means "Windows GUI" subsystem.
- 8) [Offset: 0x000001B0] **ExportTableAddress & SizeOfExportTable**: The export table address and size. The export directory contains the address of the exported functions and variables.
- 9) [Offset: 0x000001B8] **ImportTableAddress & SizeOfImportTable**: The import table address and size. The import directory contains information about functions imported from other executables. The addresses in the table are used to access the function and data from the other executable files.
- 10) [Offset: 0x00000228] **Reserved**: 8-byte zeros, you can treat it as the end of the Optional Header.

| Offset(h) | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | Decoded text     |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|------------------|
| 00000140  | 0B | 02 | 0E | 28 | 00 | 48 | 47 | 00 | 00 | 54 | 3B | 00 | 00 | 00 | 00 | 00 | ..(.HG..T;.....  |
| 00000150  | CC | 00 | 42 | 00 | 00 | 10 | 00 | 00 | 00 | 00 | 40 | 01 | 00 | 00 | 00 | 00 | İ.B.....@...     |
| 00000160  | 00 | 10 | 00 | 00 | 00 | 02 | 00 | 00 | 06 | 00 | 00 | 00 | 01 | 00 | 00 | 00 | .....            |
| 00000170  | 06 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | E0 | 82 | 00 | 00 | 04 | 00 | 00               |
| 00000180  | 56 | 7B | 82 | 00 | 02 | 00 | 60 | 81 | 00 | 00 | 10 | 00 | 00 | 00 | 00 | 00 | V{,...`.....     |
| 00000190  | 00 | 10 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 10 | 00 | 00 | 00 | 00 | 00 | .....            |
| 000001A0  | 00 | 10 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 10 | 00 | 00 | .....            |
| 000001B0  | 20 | 29 | 57 | 00 | 08 | 01 | 00 | 00 | 28 | 2A | 57 | 00 | 90 | 01 | 00 | 00 | )W.....(*W.....  |
| 000001C0  | 00 | F0 | 5A | 00 | 08 | 87 | 27 | 00 | 00 | 59 | 00 | 4C | E2 | 01 | 00 | 00 | .8Z..#'...Y.Lâ.. |
| 000001D0  | 00 | 48 | 82 | 00 | 50 | 29 | 00 | 00 | 00 | 80 | 82 | 00 | BC | 57 | 00 | 00 | .H.,P)...€,.4W.. |
| 000001E0  | 60 | CF | 51 | 00 | 1C | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | IQ.....          |
| 000001F0  | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | D1 | 51 | 00 | 28 | 00 | 00 | 00 | .....ÑQ.(...     |
| 00000200  | 20 | CE | 51 | 00 | 40 | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | IQ.€.....        |
| 00000210  | 00 | 60 | 47 | 00 | E8 | 12 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .'G.è.....       |
| 00000220  | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .....            |

In order to add a new code section, we will need to enlarge the size of the image. Therefore, we need to increase the value of **SizeOfImage** (0x830000 as an example). Remember the value of **SizeOfImage** need to be a multiple of the **SectionAlignment** value.

| Offset(h) | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | Decoded text |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--------------|
| 00000160  | 00 | 10 | 00 | 00 | 00 | 02 | 00 | 00 | 06 | 00 | 00 | 00 | 01 | 00 | 00 | 00 | .....        |
| 00000170  | 06 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 83 | 00 | 00 | 04 | 00 | 00 | .....f..     |
| 00000180  | 56 | 7B | 82 | 00 | 02 | 00 | 60 | 81 | 00 | 00 | 10 | 00 | 00 | 00 | 00 | 00 | V{,....      |

### Section Table/Section Headers (Offset: starting from 0x00000300)

After we deal with all the PE Headers, we can now start creating our own code section. Typically, the first Section Header is called ".text". Therefore, the easier way to find the start of this table is to look for this string. Each Section Header (/ Section Table Entry) has 10 fields, a total of 40 (0x28) bytes per entry. But only 6 fields are important when we try to understand a code section:

- 1) [Relative offset: 0x00] **Name**: An 8-byte, null-padded UTF-8 encoded string. Executable images do not support section names longer than 8 characters.
- 2) [Relative offset: 0x08] **VirtualSize**: The total size of the section when loaded into memory.
- 3) [Relative offset: 0x0C] **VirtualAddress**: The address of the first byte of the section relative to the image base when the section is loaded into memory. This address needs to be a multiple of **SectionAlignment** from the Optional Header (which is 0x1000 in our case).
- 4) [Relative offset: 0x10] **SizeOfRawData**: The size of the section on disk, which needs to be a multiple of **FileAlignment** from the Optional Header (which is 0x200 in our case) for executable files.
- 5) [Relative offset: 0x14] **PointerToRawData**: A pointer to the first page of the section within the file, which also must be a multiple of **FileAlignment** from the Optional Header (0x200 in our case) for executable files.
- 6) [Relative offset: 0x24] **Characteristics**: The flags that describe the characteristics of the section. Again, please check the official Microsoft documentation for more details, because it is too complicated 2.0 UwU: <https://learn.microsoft.com/en-us/windows/win32/debug/pe-format#section-flags>.

(For the remaining 3 uncovered fields, they are either already deprecated or will be set to 0 for executable images, so I won't cover them here. Just set all fields to 0 and you are good to go :D)

| Offset(h) | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | Decoded text      |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|
| 00000230  | 2E | 74 | 65 | 78 | 74 | 00 | 00 | 00 | 68 | 47 | 47 | 00 | 00 | 10 | 00 | 00 | .text...hGG....   |
| 00000240  | 00 | 48 | 47 | 00 | 00 | 04 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .HG.....          |
| 00000250  | 00 | 00 | 00 | 00 | 20 | 00 | 00 | 60 | 2E | 72 | 64 | 61 | 74 | 61 | 00 | 00 | ....`rdata..      |
| 00000260  | 60 | 08 | 10 | 00 | 00 | 60 | 47 | 00 | 00 | 0A | 10 | 00 | 00 | 4C | 47 | 00 | `...`G....LG      |
| 00000270  | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 40 | 00 | 00 | 40 | .....@...@        |
| 00000280  | 2E | 64 | 61 | 74 | 61 | 00 | 00 | 00 | 24 | 85 | 01 | 00 | 00 | 70 | 57 | 00 | .data...\$....pW. |
| 00000290  | 00 | 2E | 01 | 00 | 00 | 56 | 57 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | ....VW.....       |
| 000002A0  | 00 | 00 | 00 | 00 | 40 | 00 | 00 | C0 | 2E | 70 | 64 | 61 | 74 | 61 | 00 | 00 | ....@.pdata..     |

| Name   | VirtualSize | VirtualAddress | SizeOfRawData | PointerToRawData | Characteristics |
|--------|-------------|----------------|---------------|------------------|-----------------|
| .text  | 00 47 47 68 | 00 00 10 00    | 00 47 48 00   | 00 00 04 00      | 60 00 00 20     |
| .rdata | 00 10 08 60 | 00 47 60 00    | 00 10 0A 00   | 00 47 4C 00      | 40 00 00 40     |
| .data  | 00 01 85 24 | 00 57 70 00    | 00 01 2E 00   | 00 57 56 00      | C0 00 00 40     |

If you analyze closely, you will discover a "hidden" formula of the **VirtualAddress** and **PointerToRawData**!

**Address of current section = RoundUp( Address of previous section + Size of previous section )**

- 1) **VirtualAddress** of .data = 0x476000 (**VirtualAddress** of .rdata) + 0x100860 (**VirtualSize** of .text) = 0x576860  
and we need to round up to nearest 0x1000 (because of the **SectionAlignment**), thus, the new address = 0x577000
- 2) **PointerToRawData** of .data = 0x474C00 (**PointerToRawData** of .rdata) + 0x100A00 (**SizeOfRawData** of .rdata) = 0x575600

With this information, you can now insert a new section! Find the current last section and follow this format to insert a new section at the end of the section table. Insert extra **SizeOfRawData** null bytes into your executable based on your starting **PointerToRawData** (You may also need to add extra null bytes after the end of the previous section to reach the **PointerToRawData**), and the executable should still runnable if you insert the new section correctly. I will leave this as an exercise and release the solution in the next episode UwU (because I am running out of space...). Next episode, I will talk more about how to write a malicious code in Windows and insert that malicious code into our newly created code section, stay tuned!



# Creating Geometry Dash CTF Challenges #1 - The Basics

ivanwong13768

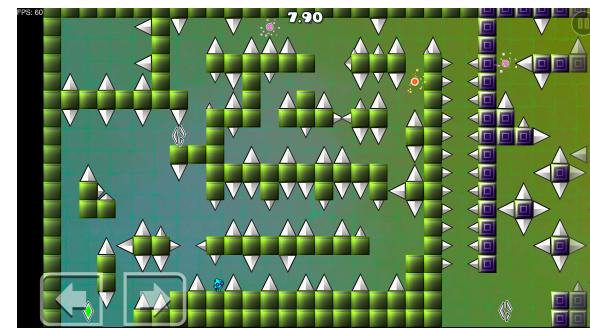
Sandbox games have been a popular medium for CTF challenges due to their high flexibility in creation and programmability. Unfortunately, Geometry Dash, a 2D platformer game known for its highly flexible level editor, is not one of them. There has only been one Geometry Dash OSINT challenge, which is disappointingly easy to solve. Geometry Dash has way more to offer than people might expect, so in this article series, I will explore the programmable side of the game and discuss ways to make CTF challenges of different categories using it.

## What is Geometry Dash?

Geometry Dash is a game about controlling your icon and navigating through a “level”. As of version 2.2, there are 2 types of levels. Usually, a level is deemed “normal” if it uses the traditional gameplay movement – the player icon automatically moves from left to right across the map, and the player has to move the icon up and down in different ways (specified by the gamemode that the icon is in) to avoid obstacles. There is also a type of “platformer” level in which the player, instead of the game, controls the icon to move left or right.



*Example of a normal level<sup>1</sup>*



*Example of a platformer level<sup>2</sup>*

Since Geometry Dash is a long name, I would simply call it “GD” throughout the series.

## Game directory structure

Before programming GD, we first have to know what the game offers. This means we have to understand the game directory and files first.

### Steam folder

GD is a Steam game, so most of the game resources are stored in the default Steam apps directory. The game directory is usually at `C:\Program Files (x86)\Steam\steamapps\common\Geometry Dash`. In case Steam is not installed in its default directory, we can find the game’s directory by opening Steam, selecting `Games` in the top left corner and navigating to Geometry Dash’s page, then clicking on the gear and choosing `Manage > Browse Local Files`.



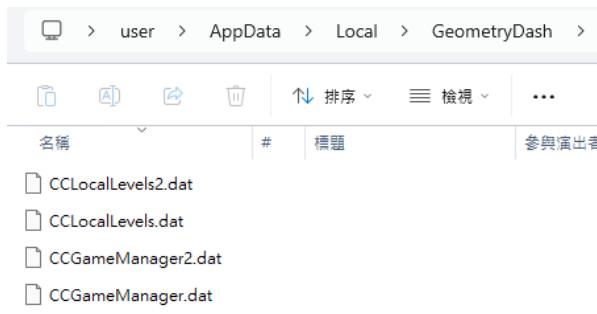
<sup>1</sup> Level: Subterranean Animism by EVW (and more)

<sup>2</sup> Level: I wanna be the guy by Aless50

The game binary, some DLLs and a **Resources** folder reside in this folder. Most files in the **Resources** folder are multimedia elements used by the game, such as game textures, main level music tracks and default sound effects. They can be replaced by the user, which is how custom texture packs are installed into the game. This is not our focus in this chapter, so let's quickly move on to the next section.

## Save data files

Client-side save data is saved in a different directory (`%localappdata%\GeometryDash` by default) along with user-downloaded custom songs. These data are stored in 2 different files, namely **CCGameManager.dat** and **CCLocalLevels.dat**.



*Not the files with 2 in their names!*

**CCGameManager.dat** stores player statistics, user-downloaded levels and game settings, while **CCLocalLevels.dat** stores user-created levels in the Create tab (aka where you access the level editor). The simplest way to view these files is [Colon's GD Save Explorer](#), but we can still manually parse them if we want to access the data programmatically. The programmatic way will be the main focus of the next section.

## **Editing save data files (the hard way)**

### Decrypting save data files

If we try to open the save data files using a text editor, we can only see many random bytes which are not exactly human-readable. This is because save data files are encrypted by a sequence of procedures for “security”. The encryption procedure is as follows:

```
gzip compress => URL-safe base64 encode => XOR all bytes with key 0xb
```

The decryption procedure is the exact reverse of the encryption procedure.

```
XOR all bytes with key 0xb => URL-safe base64 decode => gzip decompress
```

Since all of the above steps either do not use a key or use a known key, encryption and decryption can be carried out effortlessly. For example, in Python, we can simply use the **gzip** and **base64** modules to implement these 2 procedures.

Python code:<sup>3</sup>

<sup>3</sup> Modified from [GD Documentation: Game Files - Encryption and Decryption](#).

```

import gzip, base64

def xor(string: str, key: int) -> str:
    return ("").join(chr(ord(char) ^ key) for char in string)

def encrypt_data(data: str) -> str:
    gzipped = gzip.compress(data.encode())
    base64_encoded = base64.urlsafe_b64encode(gzipped)
    return xor(base64_encoded.decode(), key=11)

def decrypt_data(data: str) -> str:
    base64_decoded = base64.urlsafe_b64decode(xor(data, key=11).encode())
    decompressed = gzip.decompress(base64_decoded)
    return decompressed.decode(errors='ignore')
# errors='ignore' is added to prevent non-ASCII characters from causing errors

```

## Editing CCGGameManager.dat

Now that we have decrypted the files, let's take a look inside each of them. On the right is a very short snippet of what decrypted `CCGameManager.dat` normally contains.

```

<?xml version="1.0"?>
<plist version="1.0" gjver="2.0">
<dict>
  <k>valueKeeper</k>
  <d>
    <k>gv_0001</k>
    <s>1</s>
  </d>
</dict>

```

The file is in XML format, storing information in a large dictionary containing nested key-value pairs as entries. Most XML tags are used to represent the data type wrapped by the tag. Possible XML tags in the file include:

| Tag    | Data type <sup>4</sup>                            |
|--------|---------------------------------------------------|
| <dict> | Root dictionary                                   |
| <k>    | Key string                                        |
| <d>    | Sub-dictionary (nested inside another dictionary) |
| <s>    | String value                                      |
| <i>    | Integer value                                     |
| <r>    | Float32 value                                     |
| <t>    | True (boolean)                                    |

The keys in `CCGameManager.dat` have meaningful names, so figuring out which value to edit is fairly easy. To modify a player's local statistics and settings, simply locate the required pair in the file by searching for the key and edit its corresponding value. Some of the most important keys include:

| Key          | Data <sup>5</sup>                                      |
|--------------|--------------------------------------------------------|
| valueKeeper  | Game variables (settings), unlocked items              |
| GS_value     | Player statistics (collected stars, total attempts...) |
| GS_completed | List of completed levels                               |

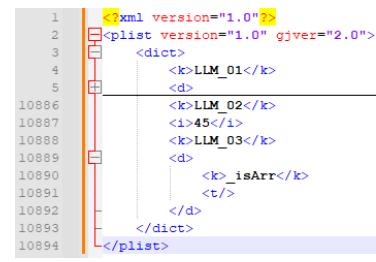
<sup>4</sup> Retrieved from [GD fandom wiki: backup of useful stuff from Save Files page](#).

<sup>5</sup> Retrieved from [GD Documentation: Client Gamesave Resource](#).

## Editing CCLocalLevels.dat

Similar to `CCGameManager.dat`, `CCLocalLevels.dat` is also in XML format. It contains 3 main entries: `LLM_01`, `LLM_02` and `LLM_03`. `LLM_01` is a dictionary of user-created levels stored in the level editor, `LLM_02` is hardcoded to be the binary version of the game, and `LLM_03` stores user-created lists of multiple levels.

Each entry in `LLM_01` stores all information of a level, not only including general elements like the level name, description, song used and all the objects inside the level, but also player-specific information like attempt count, highest percentage achieved and even jump count. Each type of information uses a distinct key, so there can be up to nearly 100 keys in an entry. However, a key would appear only if it is used, so we usually only see 20–30 keys in an entry. Some commonly seen keys include, but are not limited to:



| Key  | Data <sup>6</sup>                                              |
|------|----------------------------------------------------------------|
| k2   | Level name                                                     |
| k3   | Level description (URL-safe base64 encoded)                    |
| k4   | Compressed level string <sup>7</sup> (URL-safe base64 encoded) |
| k5   | Creator name                                                   |
| k18  | Attempt count in the level                                     |
| k104 | List of IDs of <a href="#">Newgrounds</a> songs used           |

## Useful resources in Geometry Dash programming

GD is a closed-source game and does not have built-in mod support. However, the community has reverse-engineered the game binary, written documentation and designed some tools to help interact with it. Some useful ones include, but are not limited to:

- [GD Documentation](#): Unofficial documentation written by community members about most technical details of GD. This is particularly useful when editing save files since some keys look incredibly arbitrary, as shown in the table in the previous section.
- [Awesome Geometry Dash](#): A list of GD mods, libraries, frameworks, etc. Some of them are deprecated due to the (way overdue) 2.2 update, but most remain intact.
- [JDash](#), [GeometryDashAPI](#): GD API wrappers for Java and JavaScript, respectively. ([gd.py](#) used to be a thing, but was deprecated after the developer gave up maintaining it.) These tools simplify the interaction with GD servers so we do not have to manually craft requests.
- [G.js](#): JavaScript API library for creating GD levels programmatically.
- [gdparse](#): JavaScript library for parsing GD levels and save files, then converting them into objects. Note that this library is designed for update 2.1, and its `base64.js` source file might be slightly problematic.
- [Geode](#): GD mod loader and modding SDK for C++ developers. This will be essential for writing and installing mods when we use them for CTF challenges.

In the next chapter, I will discuss some interesting ideas and relevant resources about creating GD OSINT challenges. Stay tuned!

<sup>6</sup> Retrieved from [GD Documentation: Client Level Resource](#).

<sup>7</sup> The level string represents every object inside a level. This will be covered in a later chapter.



# Speeding up Different Types of Convolutions

happypotato

In competitive programming, we might be asked to perform different types of convolutions. That is, given 2 arrays  $A, B$  of length  $n$ , calculate the array  $C$  defined by

$$C[k] = \sum_{i+j=k} A[i]B[j],$$

where  $\star$  is some binary operator. It turns out for some specific binary operators, we can calculate  $C$  in  $O(n \log n)$  time rather than the trivial  $O(n^2)$  time. In this article I will list a few useful examples and hopefully explain some of the tricks used.

## o High-level idea

For most common convolutions used in competitive programming, the optimisation follows this particular structure:

```
vector<int> convolution(vector<int> A, vector<int> B, int n) {
    vector<int> C(n);
    A = applyTransform(A);
    B = applyTransform(B);
    for (int i = 0; i < n; ++i) {
        C[i] = A[i] * B[i];
    }
    C = applyInverseTransform(C);
    return C;
}
```

In words, you are trying to find a function (transformation)  $f$  such that  $f(A) \cdot f(B) = f(C)$ . Of course, the function  $f$  differs for different operators  $\star$  used for convolution. I will list a few common examples below.

## 1 Convolution with AND, OR

We begin by tackling convolution with AND (i.e.  $i \star j = i \& j$ ). By looking at the binary decomposition, we can treat numbers as a set. For example,  $11 \rightarrow \{1, 2, 8\}$  and  $13 \rightarrow \{1, 4, 8\}$ . This way, taking AND of two numbers correspond to intersecting their sets, as  $11 \& 13 = 9 \rightarrow \{1, 2, 8\} \cap \{1, 4, 8\} = \{1, 8\}$ . Abusing notation, we will write  $C[k] = \sum_{(i \cap j)=k} A[i]B[j]$ .

Let us relax the condition by calculating the subset sum for every element instead. Define a new array  $C'$  such that  $C'[k] = \sum_{(i \cap j) \subseteq k} A[i]B[j]$ . This is good for us, as now the condition can be separated:  $(i \cap j) \subseteq k \iff (i \subseteq k) \wedge (j \subseteq k)$ . Now, if we define  $A'$  and  $B'$  similarly (so  $A'[k] = \sum_{i \subseteq k} A[i]$  and  $B'[k] = \sum_{j \subseteq k} B[j]$ ), we get  $C'[k] = A'[k]B'[k]$ ! What remains is to compute (and inverse)  $A'$  from  $A$ .

Lucky for us, this is a standard trick called Sum-Over-Subsets (SOS) DP. I won't go over that in detail here as it is literally just "contribution done wisely", but I will show that this is easily invertible. Take a look at the code for SOS DP. For simplicity, we assume that  $n$  is a perfect power of 2.

```
vector<int> applyANDTransform(vector<int> A) {
    // Also known as SOS DP
    // Assume n = 2^k
    int n = A.size(), k = floor(log2(n));
    for (int bit = 0; bit < k; bit++) {
        for (int i = 0; i < n; i++) {
            if (i & (1 << bit)) {
                A[i] += A[i ^ (1 << bit)];
            }
        }
    }
}
```

```

        }
    }
    return A;
}

```

We can easily invert this transformation as follows:

```

vector<int> applyANDInverseTransform(vector<int> A) {
    // Assume n = 2^k
    int n = A.size(), k = floor(log2(n));
    for (int bit = 0; bit < k; bit++) {
        for (int i = 0; i < n; i++) {
            if (i & (1 << bit)) {
                A[i] -= A[i] ^ (1 << bit]);
            }
        }
    }
    return A;
}

```

It is not hard to see that convolution with OR works the same, except that you take sum of supersets rather than subsets.

## 2 Convolution with GCD, LCM

Before starting on convolution with GCD, let's recap on what we did for convolution with AND: The key part of the optimisation is to relax the condition from  $C[k] = \sum_{(i \cap j)=k} A[i]B[j]$  to  $C'[k] = \sum_{(i \cap j) \subseteq k} A[i]B[j]$ . Can we do something similar for GCD and LCM?

A natural way of thinking is to relax the condition from  $\gcd(i, j) = k$  into  $k \mid \gcd(i, j)$ , which works just as above since  $k \mid \gcd(i, j) \iff (k \mid i) \wedge (k \mid j)$ . This time, finding  $A'[k] = \sum_{k \mid i} A[i]$  (equivalent to finding sum of multiples of  $k$ ) can be easily done with sieve and is  $O(n \log n)$  by harmonic sum. Also, it shouldn't be hard to observe that it is invertible. Have a look at the following code:

```

vector<int> applyGCDTransform(vector<int> A) {
    int n = A.size();
    for (int i = 1; i < n; i++) {
        for (int j = i * 2; j < n; j += i) {
            A[i] += A[j];
        }
    }
    return A;
}

vector<int> applyGCDInverseTransform(vector<int> A) {
    int n = A.size();
    for (int i = n - 1; i >= 1; i--) {
        for (int j = i * 2; j < n; j += i) {
            A[i] -= A[j];
        }
    }
    return A;
}

```

For all the operations covered above, the operator and the set of elements can be thought of as a poset. Informally, if you make a graph by adding the edge  $A \rightarrow B$  if value  $A$  is "larger" than  $B$  (e.g. in the case of GCD, if  $B \mid A$ ), the resultant graph forms a directed acyclic graph (DAG). There are also other names for the transformations; the partial sum-like transformation is called Zeta transform and the difference array-like transformation is called Möbius transform. If you are interested, there is a very good blog that explains this concept in depth (and links back to all the convolutions mentioned above).

As a final side note, you can do convolution with XOR (i.e.  $C[k] = \sum_{i+j=k} A[i]B[j]$ ) using the transform  $A'[k] = \sum_i (-1)^{|i \cap k|} A[i]$ . This and the speed-up to  $O(n \log n)$  isn't that intuitive; the transformation itself is called the Fast Walsh–Hadamard Transform (FWHT), and I believe there are other interpretations to this transform other than doing XOR-convolution, so maybe I'll write more about this in the next article.

### 3 Convolution with + (Polynomial multiplication)

Now, let's try tackling convolution with  $+$ . This is very useful, as the formula  $C[k] = \sum_{i+j=k} A[i]B[j]$  is actually equivalent to multiplying two polynomials together, which allows for the efficient use of efficient big integer multiplication or generating functions. For this reason, let's treat  $A$  and  $B$  as polynomials from now, with the coefficient of  $x^i$  being  $A[i]$  or  $B[i]$ .

First of all, note that you can recover a polynomial with degree  $n - 1$  (this means  $n$  terms) with  $n$  points. You can think of it as solving a system of linear equations with  $n$  terms, and it can be proven that the matrix has full rank. This is called interpolation, and we will try to use this fact. Here is an outline of the algorithm:

1. Pick  $n$  values  $x_0, x_1, x_2, \dots, x_{n-1}$ , where  $n = \deg(A) + \deg(B) + 1$ .
2. Convert  $A$  and  $B$  into point-value form by substituting all  $x_i$  into  $A$  and  $B$ .
3. Multiply them element by element to form a new array  $C$ .
4. Interpolate  $C$  to recover the new polynomial  $C$ .

It seems hard to improve this algorithm to  $O(n \log n)$ , but with the correct choice of  $x_i$ , it is possible! The trick is to pick  $x_i$  as the  $n$ -th roots of unity, i.e.  $x_i = \exp\left(\frac{2\pi i}{n}\right)$ . This is called the (Inverse) Fast Fourier Transform ((I)FFT), and there are multiple ways to optimise this to  $O(n \log n)$ .

The most simple one (in my opinion) is to perform divide-and-conquer on odd and even indices. Once again, we take  $n = 2^k$  for convenience. Let  $A_0(x) = \sum_{i \equiv 0 \pmod{2}} a_i x^i$ ,  $A_1(x) = \sum_{i \equiv 1 \pmod{2}} a_i x^i$ , and if  $i \geq n$  let  $a_i = 0$ . Then,  $A(x) = A_0(x^2) + xA_1(x^2)$ , which can be calculated by divide-and-conquer. It might be easier to understand this after writing the matrix (編: why matrix? Remember, linear algebra.) out, e.g. here.

How about inverting it? It turns out after some math,  $V^{-1} = \frac{1}{n}V!$  Therefore we can apply the same algorithm and dividing it by  $n$  at the very end. Sadly, I still don't have any intuition on this yet; I only know it is true. This is the final code (assume the `Complex` class is implemented):

```
// Assume n = 2^k
int n = A.size();
if (n == 1) {
    return A;
}

vector<Complex> A0, A1;
for (int i = 0; i < n; i++) {
    if (i % 2 == 0) {
        A0.push_back(A[i]);
    } else {
```

```

        A1.push_back(A[i]);
    }
}

A0 = FFT(A0);
A1 = FFT(A1);
Complex omega = root_of_unity(n);
vector<Complex> res(n);
for (int i = 0; i < n / 2; i++) {
    res[i] = A0[i] + pow(omega, i) * A1[i];
    res[i + n / 2] = A0[i] - pow(omega, i) * A1[i];
    // pow(omega, i + n / 2) = -pow(omega, i)
}
return res;
}

vector<Complex> IFFT(vector<Complex> A) {
    int n = A.size();
    A = FFT(A);
    for (int i = 0; i < n; i++) A[i] /= n;
    return A;
}

```

In competitive programming, you are usually asked to take the answer modulo some number  $M$ . This can be done by finding a  $n = 2^k$ -th root of unity  $\omega$  modulo  $M$ . The reason is that we are only using the property  $\omega^n = 1$  above during the (I)FFT steps. In competitive programming,  $M = 998244353 = 2^{23} \times 119 + 1$  is usually used, so we can take  $\omega = g^{119}$  as a  $2^{23}$ -th root of unity, where  $g$  is any primitive root (say 3). Such modification is called the Number Theoretic Transform (NTT). This can be extended further into nice rings. (編: but as Remy says, all NTT/MTT are just FFT optimisation on different rings.)

With this, you can multiply 2 polynomials efficiently and allow the use of generating functions. There are lots of tricks for that, sadly I still haven't learnt them yet, so maybe I'll write another blog afterwards.



## 前言

話說小弟近年係加國從事網絡安全相關工作，做開滲透測試，平主要就係係網站同埋手機應用程式搵下漏洞來呃啖飯食。咁小弟自問能力有限，好多時手頭份報告呢就有咩好寫。一籌莫展之際，唯有睇下一啲第三方整合應用程式有冇咩可以寫下等小弟可以同老細有個交代。於是乎有好幾次我都好好地係相關整合功能內搵到幾個頗為簡單就可以利用嘅失誤設定。我心谂嘅我咁廢都可以搵到，相信網絡上咁多大神一早已經講到爛晒。於是我就上駭客一同谷歌睇下有冇其他人有一啲類似發現，但好似冇咩人講。雖然呢啲就唔係啲咩大事，但係冇人講要我講都唔會好委屈。有見及此，小弟今日希望可以略略分享下一啲我係測試期間見開發者常用嘅第三方應用，同埋我曾經搵到啲咩。

## 對講機冇開身份認證

唔知大家有冇曾經見有啲網站有個信使俾你可能同客服傾計或者問問題。其中一個幾常見嘅提供者——對講機。咁樣話說我就試緊一個軟體式服務平臺，就唔開名啦，用對講機比你同佢地啲職員吹水。事關我留意到係個對話方塊入面個系統好似認到我個電郵地址，又可以睇返我以前同職員啲對話記錄，咁我就好似平時咁樣睇下中間人代理啲代理歷史睇下到底呢個對講機到底係點樣認邊個打邊個啦。一睇原來個生請求入面又有曲奇，標頭又有授權令牌，到底係點樣知道我係邊個呢？我就發現原來個對講機端點只需要係個請求身體內提供應用標識號同埋係使用者資訊提供我對應嘅電郵地址，我就可以擢到我自己啲對話記錄。個應用識別碼就係一個應用共用同一個識別碼，我能夠改變嘅就只有個電郵啦。咁樣好合理地我下一樣野就試下其他電郵地址啦，發現原來只要我有其他用戶嘅電郵地址，我就可以擢到人地個對話記錄，又可以冒充個受害者同個客服對話。我心谂冇理由咁巒居啊，就走去對講機個應用程式介面文檔到睇下到底係咪真係咁樣認證用戶。原來呢就真係冇咁巒居。對講機有個功能叫身份認證，就係會由伺服器用一個伺服器端嘅秘密同用戶電郵地址或者用戶識別號簽發一個金鑰雜湊訊息鑑別碼，以此鑑別碼認證用戶。呢個功能係可以被停用，而係個官方文檔度亦都大大隻字建議你開個功能。如下圖示。

[Home](#) / [Web](#) / [Identity Verification](#)

## Identity Verification

Identity Verification helps to make sure that conversations between you and your users are kept private and that one user can't impersonate another. **We strongly encourage all Intercom customers to set up and enable Identity Verification.**

Identity Verification works by using a server side generated [HMAC \(hash based message authentication code\)](#), using SHA256, on either the user's `user_id` or `email`.

換言之我個客可能開發完左之後唔記得開返個功能，結果呢我就可以多一樣野寫啦。為左可以畀諸位讀者有實質得益，以下為我提供嘅概念認證，其功能主要為證明身份認證冇開，可以直接用電郵地址讀取用戶聊天記錄：

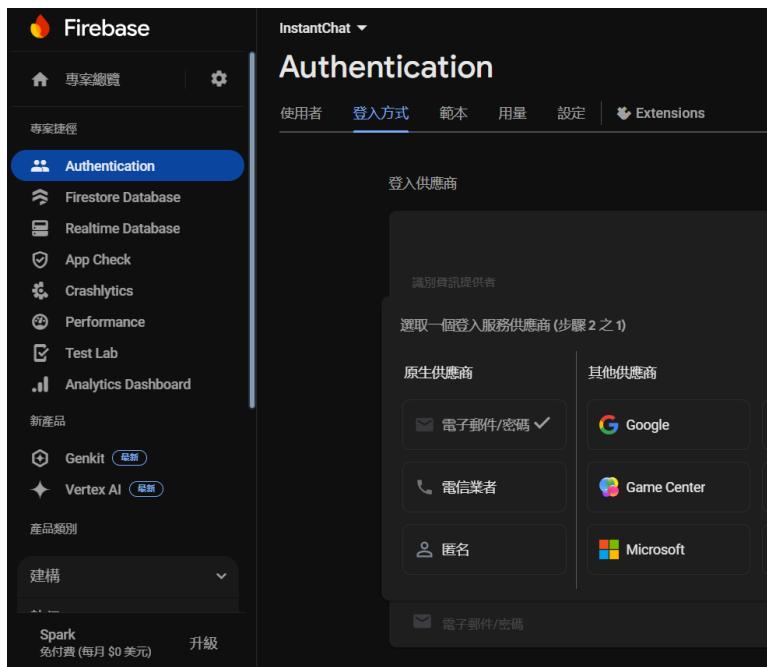
```
import requests
EMAIL = "tomchan@gmail.com"
session = requests.Session()
burp0_url = "https://api-iam.intercom.io:443/messenger/web/home"
session.headers = {"Content-Type": "application/x-www-form-urlencoded"}
def readMessage(email, app_id="abcd1234"):
    burp0_data = {"app_id": app_id, "user_data": f"{{\"email\": \"{email}\", \"app_id\": {app_id}}}"}
    res = session.post(burp0_url, data=burp0_data)
    if(res.status_code == 200):
        print(f"[{res.json()['cards'][1]['conversations'][0]['conversation_message']}]")
readMessage(EMAIL)
```

# 火基地俾你用電郵密碼注冊

相信大家對火基地都唔陌生，佢係一個由谷歌提供後端服務平台。其中有各樣服務，例如認證同埋資料庫。一般來講，用戶自行注冊係一個幾常見嘅用例，但係某啲特定嘅應用可能有特定商業需求要限制用戶注冊。

咁話說我當時就係試緊一個投資應用。一般用戶注冊就要填寫一張表去提供用戶資料以驗證身份，例如上傳身份證明文件同埋驗證電話號碼等，先批准一個用戶係平臺注冊並使用相關服務。咁我就留意到係我個中間人代理歷史入面主要認證用戶嘅端點都係用谷歌身份平臺。係谷歌文檔當中留意到，除左手機短訊之外仲有其他認證方法，例如匿名同埋電子郵件/密碼。

如下圖示，原生供應商有三種注冊方式：



當然匿名注冊就係當時試就唔得啦，但係個電子郵件/密碼反而就得。一般來講可能係個客開來做測試，試完之後就唔記得刪返。結果咁樣我就可以唔需要一個有效電話號碼去註冊一個投資賬戶。再者，由於呢個並非預期嘅注冊途徑，所以連電郵驗證都唔需要做就可以有一個有效工作階段。不過當時試嘅時候除左可以繞過短訊驗證同埋用任何電子郵件地址註冊之外，對個系統或者其他用戶好似都有咩特別影響，所以本人憑感覺評估實際網絡安全風險唔高，最多係違反相關金融規例，如認識你嘅客戶。

小弟亦曾經將呢個發現包裝做一個奪旗挑戰並於二零二三年嘅黑荊奪旗比賽中出現。如果你有興趣你應該仲可以玩到個挑戰。

<https://github.com/blackb6a/blackb6a-ctf-2023-challenges/tree/main/15-instant-chat>

## 結論

其實小弟仲有其他幾個有趣發現，不過礙於篇幅所限，就留返下次再寫。正所謂人係網絡安全入面最脆弱嘅一環，只要某一樣野需要人去操作就會有機會出錯，而好多時第三方整合工具就係會咁。雖然呢唔係乜驚天地泣鬼神嘅零時差漏洞，但正正因為佢地夠簡單常見，更值得大家留意。望周知。



# How to plagiarize challenges for HKCERT CTF

## Mystiz

This is the fifth year that Black Bauhinia helped HKCERT to host their annual CTF. In particular, I developed 11 series summing up to 18 challenges in total. Remarkably, most of them are unoriginal and the idea comes from somewhere else. Below is the challenge description for **Mask-Mask-RSA**, the hardest crypto challenge in the qualification round.

自 2020 年起，Mystiz 開始以抄襲自己嘅題目變得有名。

到今時今日，情況從糟糕變得難以理解。佢依家開始抄其他 CTF 嘅題目。更加痴線嘅係，佢知道條問題嘅原作者係會玩呢場 CTF 嘅。

利申：個原作者喺見到呢條問題之前都唔知我抄佢。

Guess when the first idea was created? The answer is August 28, 2019 – while my former colleague, *harrier*, was working on a Javascript package together. We were debugging and found out that `Buffer.from` truncates when there is an odd number of hex characters. As a sadist, we would make more people suffer if an engineering bug kept us struggling. I also found that `Buffer.from` is also truncating *silently* if a character is not hexadecimal. Denote the user's input by `payload` and `payloadBuffer = Buffer.from(payload, 'hex')`. It is a vulnerability if we are validating with `payloadBuffer` while using `payload` to craft a SQL query. Finally here comes **Mask-Mask-RSA**, a 450-point web challenge<sup>1</sup>!



Samuel 23:12

個 3 倆人食左落肚



harrier 23:12

The vulnerabilities from **Mystiz's Mini CTF** series is also based on real-life. The vulnerabilities, grouping unintended attributes & mass assignment, respectively come from various pentests that I performed in these few years. The moral of this series is to properly limit the attributes users can use. A contestant identified a mass assignment vulnerability on their web app by looking at this challenge, which made me quite happy.

Remember the challenge description in **Custom Web Server**? We are not making the quote up. It comes from the infamous Facebook group, *the Hong Kong Programming Society*<sup>2</sup>, where the owner claimed that he made a web framework by himself. I had a brief read on the open-sourced repository and found that the framework is pretty vulnerable<sup>3</sup>. I was then very motivated to write a framework myself to show that this is a bad idea to reinvent the wheel if you have skill issues.

Mystiz's Fan Club Should Be 蔊'd 2024/2/29 上午9:02  
如何自家製 server 而霸氣全失

Chan Mc  
自家製server有個好處是被hack 難度大增。  
40分鐘 讀好 回覆

上午9:03 nginx + handcraft server, http request smuggling

Unsurprisingly, I also write reverse engineering challenges. While I was playing **NuttyShell CTF** earlier this year, I came across a `.pyc` reversing challenge written by *Ricky*. Although the intended solution is to use a disassembler to recover Python's bytecode from the binary, I noticed that there are several imports like `random` and `datetime`. Instead of reading the Python bytecode, I decided to

<sup>1</sup> The intended solution is to deposit 10 ETH to my account. Dealing with SQL injection is totally unexpected.

<sup>2</sup> <https://www.facebook.com/groups/hongkongprogrammingsociety/posts/1104651610725253/>

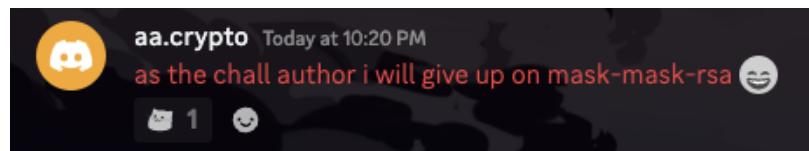
<sup>3</sup> Well, disclaimer: *Please note that most of the attacks would be ILLEGAL if attempted on machines that you do not have explicit permission to test and attack. I assume no responsibility for any actions performed outside.*

create several files like `random.py`. In this case, instead of the `random` standard library, we are importing the functions from `random.py`. Driven by the idea, I wrote **Cyp.ress** (*harrier* implemented the anti-decompilation part) - where it is preferred to override files.

I am a lone wolf when it comes to developing the cryptography challenges, and worked on most of the challenges in the category. *hoisanrd* threw me the idea for **mAEstro** (1), where I quickly came up with its sequel, **mAEstro** (2). Guess I am even plagiarizing my teammate now. Anyways, what I think is hilarious are the two avatars in the series<sup>4</sup>.



I borrowed ideas from various sources, and it seemed that I still had originality making my own challenges. One undeniable evidence of plagiarism being **Mask-Mask-RSA**. I have been dreaming of creating challenges based on the contestants' activity. For instance, last year, I read *Ricky*'s open-source coursework and investigated whether he has hashed passwords properly. Eventually, I copied the challenge from `aa.crypto` in *CyberSpace CTF 2024*<sup>5</sup> and made minimal changes – I just changed `e = 3` to `e = 65537` and reduced the number of user-provided expressions from 20 to 6. The below screenshot comes from *hollow*, where he captured the moment when `aa.crypto` rage quits.



The most original series is **RSA LCG**, where the primes are obtained from a linear congruential generator. There are four challenges in this series, and they are intended to solve in a completely different way. However, I wanted to release five parts at the beginning, with the final part having LCG unconstrained, i.e., `lcg = LCG(bits=256)`. I did not release it simply because I cannot solve that, and I do not want to become *factoreal*. If you are interested, feel free to look into this challenge!

I found myself quite productive when creating challenges this year. I surely am unable to tell others on how to make great challenges, but ideas come from everywhere. Observe and you might be finding some gems.



<sup>4</sup> Disclaimer: I did not make any of the avatars.

<sup>5</sup> <https://github.com/OfficialCyberSpace/CSCTF-2024/tree/main/crypto/mask-rsa>



## Context

For those who haven't heard of our co-learning/study week initiative, it's an experimental project launched by (mostly) the core members, with the aim of replicating the DeFi intensive co-learning project. The original aim of the project is to force participants to complete a CTF challenge every day in an unfamiliar category, particularly fueled by the psychology of peer pressure, as the participants' progress is put on display for everyone to look at.

Putting our members' busy schedules into consideration, we have decided to do a smaller scale version—with members preparing sets of challenges in rotating categories every week, and small amounts of incentives to motivate players. To test out the concept, the author of this article was assigned to pick a set of Web challenges of varying difficulty—to do a “trial” week.

## Problem Selection

I chose 9 easy 6 medium and 3 hard Web challenges for the trial week, in hopes that players without much Web experience will find at least a few challenges that they can attempt, while more advanced players can entertain themselves with the harder challenges. With regret, I have to say that I now consider this approach to be somewhat of a blunder—players without much CTF experience weren't able to enjoy the relatively easy challenges, and more experienced Web players simply weren't all that interested to justify putting in the more difficult challenges.

If I were to pick the challenges again, I would aim to put more actually easy challenges in the set, such as HKCERT CTF challenges with a difficulty rating of 1, or pair up challenges which beginners might find challenging with “step-by-step” guides—ones which show the author's approach (including googling) instead of simply listing out the steps. With categories such as Web that require a very diverse set of skills, it would perhaps be interesting to see how a “challenge tree” would play out, with each branch being a family of techniques (or maybe adding challenge tags can achieve the same effect).

## Platform

Fearing that simply putting the challenges on discord would make them too unattractive, I (or rather, ChatGPT) also built a [website](<https://chall.mov/>) in addition to modifications to the discord bot. Participants were able to submit write ups and “flag” challenges through discord, plus look at the challenges and scoreboard on the website.

Overall, ChatGPT did a good job and the only improvements that I would make next time around would be to add a write up submission feature and host challenges for players too.

## The Co-Learning Experience

To my surprise, a decent amount of members from the discord server participated in the trial week. Notably, there were 13 players who completed and submitted at least one challenge and 8 players with more than 4 tasks. Yet among those 8 players, only 1 of them (*WarriorX*) is not already a member—way less than what I had expected, especially when compared to the amount of people joining the introduction session. (*Perhaps people aren't all that interested in Web...*)

Significantly, it's particularly exciting to see members who specialise in other categories explore different aspects of Web in terms of CTF, and even more encouraging to see that they were motivated to share and exchange ideas, learning from others' approaches while sharing their own.

Speaking of sharing approaches, the write up centered aspect of this experiment has cultivated some of the most detailed yet easy to follow solution guides I've seen as a CTF player. This core idea facilitated the swapping of ideas really well, producing multiple guides per challenge and letting beginners read and follow the one they are most comfortable with. Having that said, the implementation of this with discord channel threads probably wasn't the best idea, whereas an integrated version on the website would be much more ideal.

## Conclusion

All and all the co-learning project went quite well for what supposedly is a “trial week”, yet the participation levels from non-members is somewhat worrying. Hopefully this can be resolved with more mature iterations and better difficulty scaling!

| # | Member                                                                                                                  | Solves | Score |
|---|-------------------------------------------------------------------------------------------------------------------------|--------|-------|
| 1 |  a1668k<br>Member since 16/11/2022   | 10     | 1300  |
| 2 |  WarriorX<br>Member since 18/09/2024 | 8      | 1000  |
| 3 |  Eason<br>Member since 08/09/2024    | 7      | 1000  |
| 4 |  Kaiziron<br>Member since 05/10/2020 | 8      | 1000  |
| 5 |  Viky<br>Member since 27/02/2021     | 7      | 800   |

# *Credits and Afterwards*

- **Editor-in-chief:** *GonJK*

- **Article contributors:**

*a1668k*

*GonJK*

*Mystiz*

*botton*

*grhkm*

*Ozetta*

*Eason*

*happypotato*

*Starry Miracle*

*ensy*

*ivanwong13768*

*vikychoi*

- **Design:** *a1668k & ringo*

- **Cover art:** *GonJK*

- **Article review (Knowledge-wise):**

*GonJK*

*grhkm*

*harrier*

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>