Android-PIN-Bruteforce

Unlock an Android phone (or device) by bruteforcing the lock screen PIN.

Shashi Kant Rocky
20MCMI05

V Anvesh Raju
20MCMI19

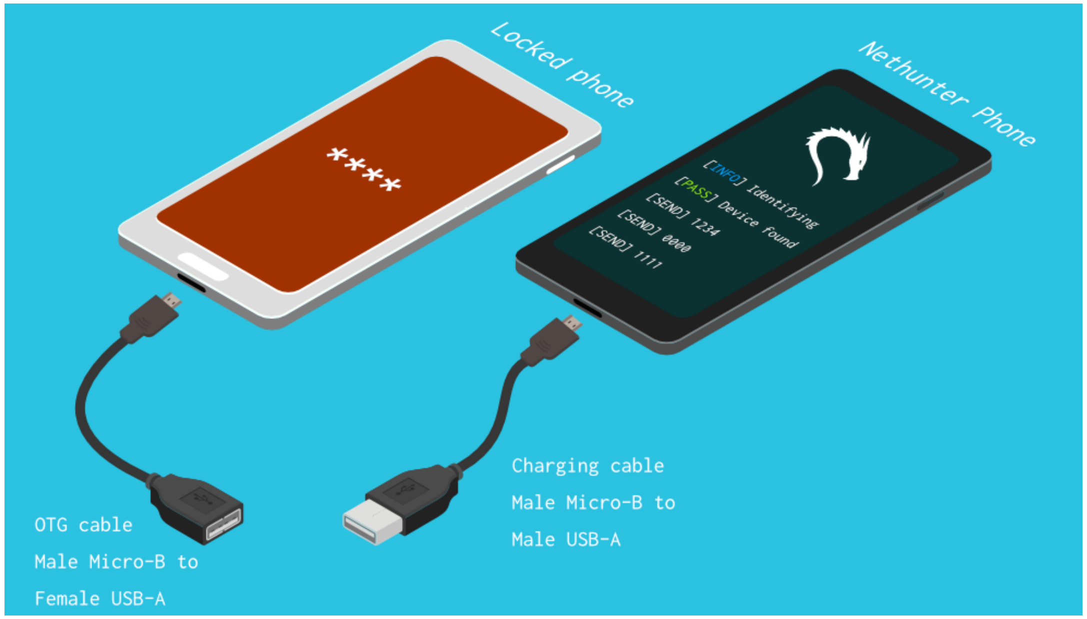
B Upendra
20MCMI22

```
:/data/local/tmp/Android-PIN-Bruteforce
ruteforce crack --length 4
Android PIN brute-force :: version 0.1
[     ] Loading optimised PIN list for 4
 digits (optimised-pin-length-4.txt)
[     ] PIN list contains 10000 PINs
[     ] Checking environment
[PASS] HID device (/dev/hidg0) found
[PASS] hid-keyboard executable (/data/l
ocal/tmp/hid-keyboard) found
```

Android-PIN-Bruteforce

Introduction

Unlocks an Android phone (or device) by bruteforcing the lock screen PIN.
Turn our Kali Nethunter phone into a bruteforce PIN cracker for Android devices!



```
[INFO] Identifying
[PASS] Device found
[SEND] 1234
[SEND] 0000
[SEND] 1111
```

It uses a USB OTG cable to connect the locked phone to the Nethunter device. It emulates a keyboard, automatically tries PINs, and waits after trying too many wrong guesses.

[Nethunter phone] <--> [USB cable] <--> [USB OTG adaptor] <--> [Locked Android phone]

Android-PIN-Bruteforce

Purpose

- Forgot Pin
- Found old Smartphone : Found old smartphone and would like to retrieve data that are stored but don't recall the pin code
- Unlock deceased Android
- Forensic Analysis

Android-PIN-Bruteforce

Few Unlock Options

- Exploit : Exploit that bypasses lock screen protection
- Using Trail and Error : Which could be automated into a brute force
- Factory data reset : If the data is not so important or we don’t care about the data

Thus as we can see , there are not many options so we need to find an efficient method to brute force .

Android-PIN-Bruteforce

Limitations

Using brute force has its limitations

- **Timeout** : There is no way to bypass timeout without exploit
    - After 5 wrong PINs – 30 seconds cooldown
    - Another 5 wrong PINs – 30 seconds cooldown
    - After 1 wrong PIN – 30 seconds cooldown
    - After 41 wrong PIN – 60 seconds cooldown
- **Time to brute force PIN** : If we put this in actual numbers how long would it take to unlock a different PIN protection.
    - 4 digits - 167 hours(7 days)
    - 5 digits – 1666 hours(69 days)
    - 6 digits -16666 hours(2 years)
    - 7 digits – 166666 hours(19 years)
    - 8 digits – 1666666(190 years)

    Thus by increasing the number of digits of PIN , the time spent on cracking exponentially increases
- **Factory reset** : Not set by default but if the user sets its from settings it means that after 15 wrong attempts the device will erase all the data

Considering stock Android . Custom UIs may have some differences.

Android-PIN-Bruteforce

Brute force can be performed using two options :

- ADB (USB debugging) : where USB debugging needs to be enabled
- HID (Rubber Ducky) : Human interface device Rubber Ducky

ADB in most cases is not really useful. HID(Rubber Ducky) is the preferrable method here .

Why

HID

is preferred?

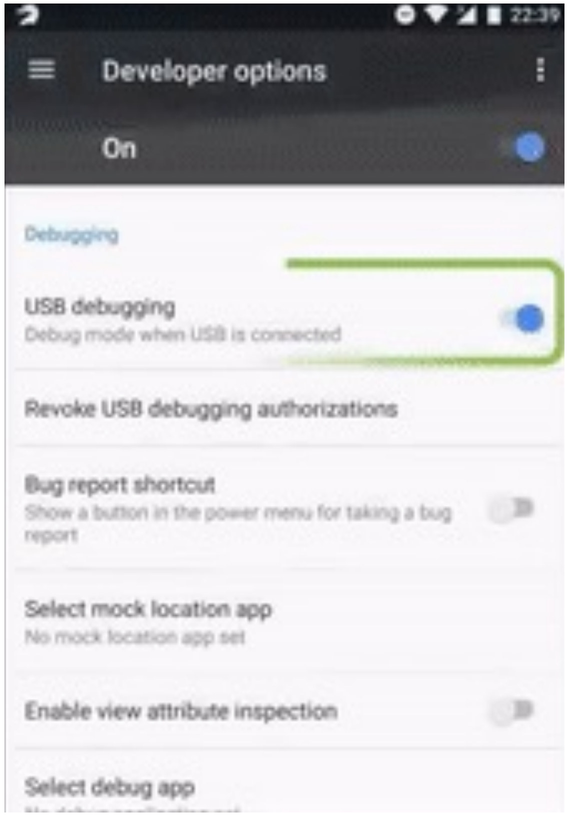
Brute force-ADB

- Requirements :
  - USB debugging enabled
  - Device authorized
  - PC or Termix

The problem is that if we receive a smartphone that is already locked , there is no way to enable USB debugging and authorized and authorized smartphone. Thus, this method is not really useful.

TOOL :

- WBRUTER(https://github.com/wuseman/WBRUTER)
  - Bypassing timeout
  - on Android 8.0 (https://www.xda-developers.com/android-oreo-programmatically-change-lockscreen-pin-password-pattern/)


22:39

Developer options

On

Debugging

USB debugging
Debug mode when USB is connected

Revoke USB debugging authorizations

Bug report shortcut
Show a button in the power menu for taking a bug report

Select mock location app
No mock location app set

Enable view attribute inspection

Select debug app

Using ABD there is no way to bypass timeouts except for one Android version 8.0 , which introduced a new feature , which made it possible to change pin, pattern , password , lock screen using new ADB command and that feature contained a bug which would result in bypassing timeouts but only for Android 8.0. However, this bug is not present anymore

Android-PIN-Bruteforce

Brute force-ADB

WBRUTER

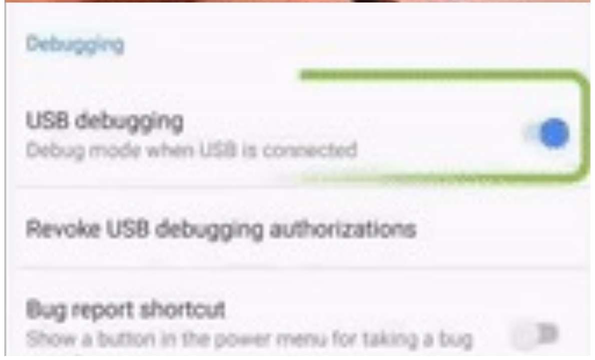
Example of bypassing timeouts and why it is not really helpful.

WBRUTER goes through all the pin combinations starting from 000 (for 4-digit PIN) until it reaches the desired input which is the actual password.

There are no throttles on the pin but we can see the errors which means there should be timeout cooldown but this is because of the bug which was not implemented correctly in Android 8.0 which is now fixed already

Using this we can unlock our smartphones using all the combinations in a short time period.

However , we still need USB debugging enabled and device authorization and again other versions of android (even the updated version of android 8.0) are not supported , which makes this technique obsolete.



Debugging

USB debugging
Debug mode when USB is connected

Revoke USB debugging authorizations

Bug report shortcut
Show a button in the power menu for taking a bug

HID

(Rubber Ducky)

using

Android-PIN-Bruteforce

Android-PIN-Bruteforce

Brute force-HID

- Requirements :
  - A Nethunter phone or **any rooted Android phone with enabled HID** or Rubber Ducky USB(android with HID Kernel Support)
  - **USB OTG** (On The Go) cable/adapter (USB male Micro-B to female USB A), and a standard charging cable (USB male Micro-B to male A)
  - **Termux**

TOOL for HID :

- **Android-PIN-Bruteforce**
- (https://github.com/urbanadventurer/Android-PIN-Bruteforce)

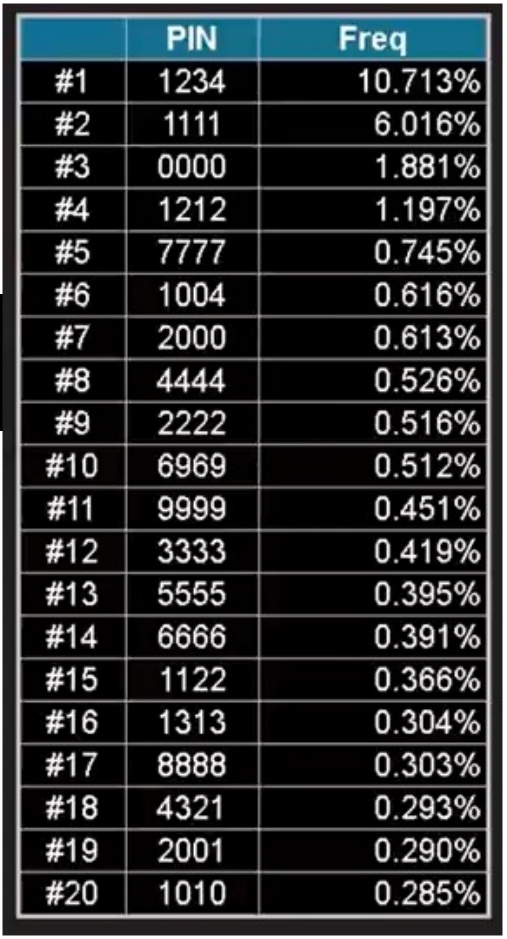
The ability to perform a bruteforce attack doesn't depend on the Android version in use. It depends on how the device vendor developed their own lock screen.

Android-PIN-Bruteforce

Pros & Cons :

- Crack PINs of any length from 1 to 10 digits
- Use config files to support different phones
- Optimized PIN lists for 3,4,5, and 6 digit PINs(Based on top-most popular pins being used based in statistics)
- Bypasses phone pop-ups including the Low Power warning
- Detects all timeouts
- Detects when the phone is unplugged or powered off, and waits while retrying every 5 seconds
- Configurable delays of N seconds after every X PIN attempts
- Log file

- Can’t log correct PIN , continues guessing : its HID connected keyboard in this case , it cannot receive any events from a smartphone so t doesn't know whether the pin was correct or not. So it would continue guessing and entering other pins even when the pin correct pin is found. Because of that we need to manually grab the attempt from the logs and compare it with a pin that is in a pin list.



	PIN	Freq
#1	1234	10.713%
#2	1111	6.016%
#3	0000	1.881%
#4	1212	1.197%
#5	7777	0.745%
#6	1004	0.616%
#7	2000	0.613%
#8	4444	0.526%
#9	2222	0.516%
#10	6969	0.512%
#11	9999	0.451%
#12	3333	0.419%
#13	5555	0.395%
#14	6666	0.391%
#15	1122	0.366%
#16	1313	0.304%
#17	8888	0.303%
#18	4321	0.293%
#19	2001	0.290%
#20	1010	0.285%

Android-PIN-Bruteforce

Where did these optimized pins come from?

The optimized PIN lists were generated by extracting numeric passwords from database leaks then sorting by frequency. All PINs that did not appear in the password leaks were appended to the list.
The optimized PIN lists were generated from Ga$$Pacc DB Leak (21GB decompressed, 688M Accounts, 243 Databases, 138920 numeric passwords).

The 4 digit PIN list

The reason that the 4 digit PIN list is used from a different source is because it gives better results than the generated list from Ga$$Pacc DB Leak.
optimised-pin-length-4.txt is an optimized list of all possible 4 digit PINs, sorted by order of likelihood. It can be found with the
filename pinlist.txt at https://github.com/mandatoryprogrammer/droidbrute
This list is used with permission from Justin Engler & Paul Vines from Senior Security Engineer, iSEC Partners, and was used in their Defcon talk, [Electromechanical PIN Cracking with Robotic Reconfigurable Button Basher (and C3BO)]()

Android-PIN-Bruteforce

Installation and Execution :

Just Install :

```
# git clone https://github.com/urbanadventurer/Android-PIN-Bruteforce.git
```

Change mode for execution :

```
#chmod + x android-pin-bruteforce
```

Trigger it :

```
./android-pin-bruteforce
```

There are various options. Let's say we want to crack 4-digit pin then this is the command :

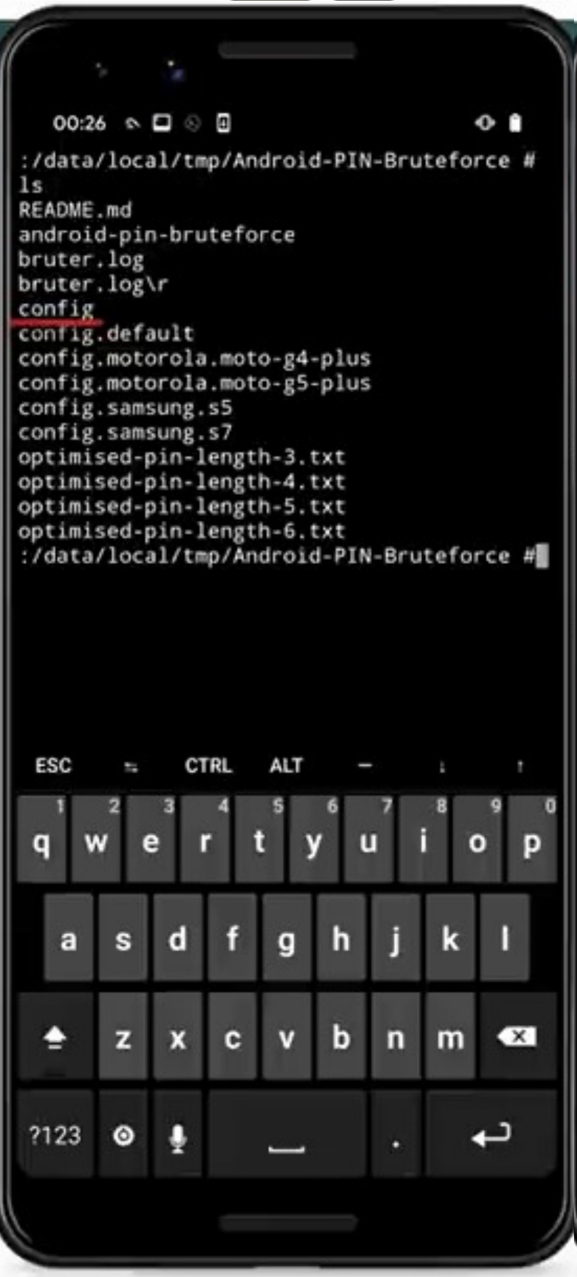
```
# ./android-pin-bruteforce crack –length 4
```

Use the --length command line option.

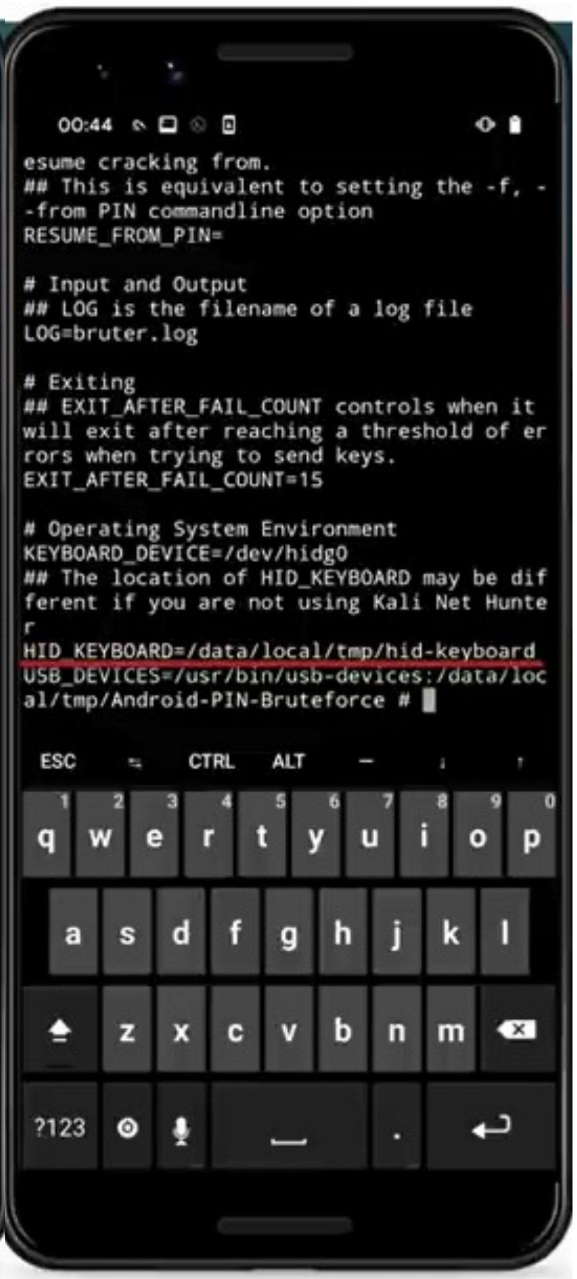
Use this command to crack a 3 digit PIN, ./android-pin-bruteforce crack --length 3

Use this command to crack a 6 digit PIN ./android-pin-bruteforce crack --length 6

Android-PIN-Bruteforce



```
00:26
:/data/local/tmp/Android-PIN-Bruteforce #
ls
README.md
android-pin-bruteforce
bruter.log
bruter.log\r
config
config.default
config.motorola.moto-g4-plus
config.motorola.moto-g5-plus
config.samsung.s5
config.samsung.s7
optimised-pin-length-3.txt
optimised-pin-length-4.txt
optimised-pin-length-5.txt
optimised-pin-length-6.txt
:/data/local/tmp/Android-PIN-Bruteforce #
```



```
00:44
esume cracking from.
## This is equivalent to setting the -f, -
-from PIN commandline option
RESUME_FROM_PIN=

# Input and Output
## LOG is the filename of a log file
LOG=bruter.log

# Exiting
## EXIT_AFTER_FAIL_COUNT controls when it
will exit after reaching a threshold of er
rors when trying to send keys.
EXIT_AFTER_FAIL_COUNT=15

# Operating System Environment
KEYBOARD_DEVICE=/dev/hidg0
## The location of HID_KEYBOARD may be dif
ferent if you are not using Kali Net Hunte
r
HID_KEYBOARD=/data/local/tmp/hid-keyboard
USB_DEVICES=/usr/bin/usb-devices:/data/loc
al/tmp/Android-PIN-Bruteforce #
```

After we get clone its necessary to edit config file that contains in one of the last lines hid keyboard global value that has a path to hid binary for the keyboard

This needs to be replaced for each device.

Cracking with Masks :

Masks use regular expressions with the standard grep extended format.

```
./android-pin-bruteforce crack --mask "...[45]" --dry-run
```

- To try all years from 1900 to 1999, use a mask of 19..
- To try PINs that have a 1 in the first digit, and a 1 in the last digit, use a mask of 1..1
- To try PINs that end in 4 or 5, use ...[45]

Android-PIN-Bruteforce

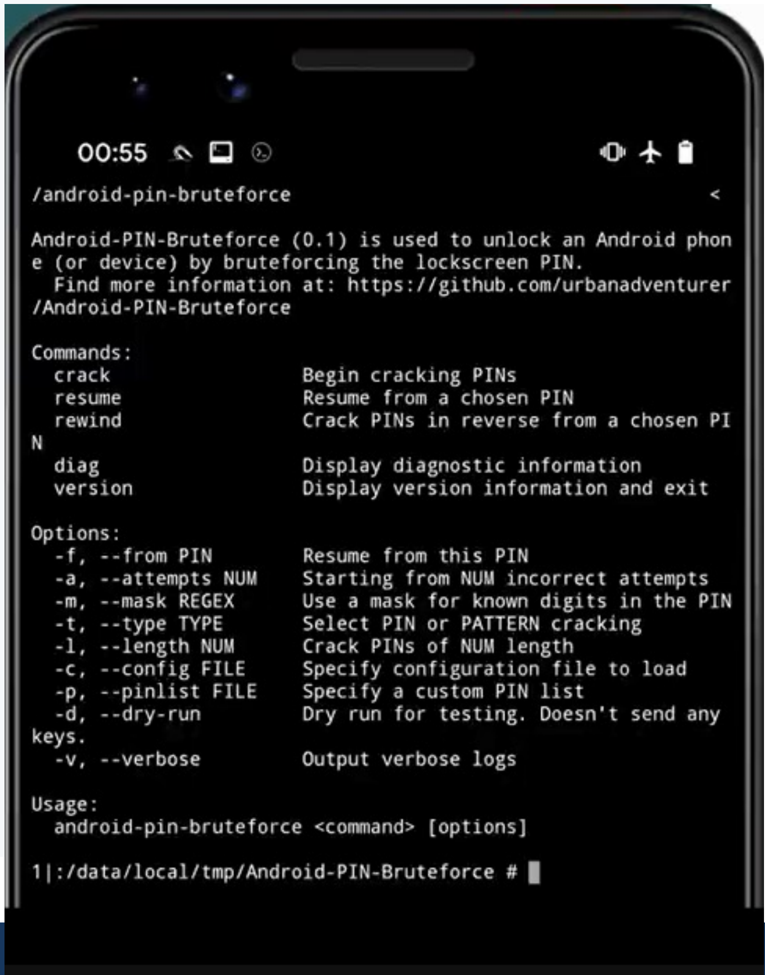
Usage

```
Android-PIN-Bruteforce (0.1) is used to unlock an Android phone (or device) by bruteforcing the lockscreen PIN
  Find more information at: https://github.com/urbanadventurer/Android-PIN-Bruteforce

Commands:
  crack                    Begin cracking PINs
  resume                   Resume from a chosen PIN
  rewind                   Crack PINs in reverse from a chosen PIN
  diag                     Display diagnostic information
  version                  Display version information and exit

Options:
  -f, --from PIN           Resume from this PIN
  -a, --attempts           Starting from NUM incorrect attempts
  -m, --mask REGEX         Use a mask for known digits in the PIN
  -t, --type TYPE          Select PIN or PATTERN cracking
  -l, --length NUM         Crack PINs of NUM length
  -c, --config FILE        Specify configuration file to load
  -p, --pinlist FILE       Specify a custom PIN list
  -d, --dry-run            Dry run for testing. Doesn't send any keys.
  -v, --verbose            Output verbose logs

Usage:
  android-pin-bruteforce <command> [options]
```



```
00:55
/android-pin-bruteforce                                        <

Android-PIN-Bruteforce (0.1) is used to unlock an Android phon
e (or device) by bruteforcing the lockscreen PIN.
  Find more information at: https://github.com/urbanadventurer
/Android-PIN-Bruteforce

Commands:
  crack                     Begin cracking PINs
  resume                    Resume from a chosen PIN
  rewind                    Crack PINs in reverse from a chosen PI
N
  diag                      Display diagnostic information
  version                   Display version information and exit

Options:
  -f, --from PIN            Resume from this PIN
  -a, --attempts NUM        Starting from NUM incorrect attempts
  -m, --mask REGEX          Use a mask for known digits in the PIN
  -t, --type TYPE           Select PIN or PATTERN cracking
  -l, --length NUM          Crack PINs of NUM length
  -c, --config FILE         Specify configuration file to load
  -p, --pinlist FILE        Specify a custom PIN list
  -d, --dry-run             Dry run for testing. Doesn't send any
keys.
  -v, --verbose             Output verbose logs

Usage:
  android-pin-bruteforce <command> [options]

1|:/data/local/tmp/Android-PIN-Bruteforce # █
```

Android-PIN-Bruteforce

Demonstration

Here’s a quick example :
The main smartphone executes the script where we try to guess the pin with 4-digits.

After 5 attempts there is a time-out and the script waits until time-out.

In this case it went through 44 pins. First five attempts then time-out. Another five attempts then again time out.

When we reach attempt 41 the timeout would increase to 60 seconds.

It continues until we reach attempt 44 when we guess the correct pin and as we can see it doesn't stop after guessing the correct pin . It continues with attempt number 45 and further.

```
:/data/local/tmp/Android-PIN-Bruteforce
ruteforce crack --length 4
```

Based on this number of attempt we have to search the pin in the list of optimized pins and we also see the time how long it took which in this case was around 25 minutes

Prevention

- **If you think you might be a target then :**
  - Use longer PINS (6,8 digits) as it will take years to crack
  - Don’t use easy to guess pin codes. One can also switch to passwords
  - Set factory reset (upto 15 wrong attempts). However this is not recommended.

Conclusion

- **Locked Smartphones** : Not many possibilities
    When we have a locked smartphone there are not many possibilities how to unlock such smartphone. We need to either have an exploit or we need to have to brute force the pin. There is no other option.

- **This Technique only works against PINs.**
    No passwords since it maybe more complex . We would need a wordless tracking pattern which is not that simple because our script needs to be customized for various displays and need to perform clicks and swipes at particular parts of the screen. Also using password is much more secure.

- **Using brute force is a long lasting operation** but there is no other option how to unlock such protected smartphone.

Bibliography

- WBRUTER: https://github.com/wuseman/WBRUTER
- Android PIN Bruteforce: https://github.com/urbanadventurer/Android-PIN-Bruteforce
- Android 8.0 update + issue : https://www.xda-developers.com/android-oreo-programmatically-change-lockscreen-pin-password-pattern/
- How to use Android as Rubber Ducky with NetHunter - https://youtu.be/bYfict-752k
- How to setup Android as Rubber Ducky without NetHunter - https://youtu.be/Mek9DMGy8os