A Brute-Force System for HID (Human Interface Device) Attack for Android Lock Screen Devices with Automatic Unlock Detection

Mateus Nunes[1], **Rafael Schneider**[1]

[1]Diretoria de Criminalística – Polícia Científica de Santa Catarina (PCI-SC)
Av. Governador Ivo Silveira, 1521 – 88085-002 – Florianópolis – SC – Brazil

{mateus.nunes,rafael.schneider}@policiacientifica.sc.gov.br

Abstract. *Electronic devices can be used to conduct illegal activities making examination of their contents essential for criminal investigation. Many mechanisms implemented over the years for securing users' privacy can hinder lawful access to phone data. This research investigates attack via USB-OTG (On-the-Go) using HID (human interface device) and introduces a system for both PIN and pattern screen lock brute-force. Contributions include stop/resume attack and automatic unlock detection via sensor on display. An assessment was done with 18 compatible devices from real cases and found that 66% were compatible with password retrieval, most of them in less than one week.*

1. Introduction

With the increasing role of electronic devices in everyone's daily life and therefore growing demand for privacy and security, lawful access to electronic devices has become a challenge. Accordingly to [Reedy 2020], we should expect on the next years an increase in number of cybercrimes involving IoT (Internet of Things) devices, online banking, virtual currencies, fraud, among other areas, and it is reported that the digital forensics worldwide market will grow from $4.62B in 2017 to $9.68B by 2022.

Digital evidences forensically acquired can be of great concern in many criminal cases given its potential in establishing impartial factual data that can acquit an innocent or condemn a guilty criminal. These evidences encompass crucial artifacts extracted from device's memory include: list of contacts, call history, text messages received and sent, pictures and videos, and GPS location data [Chernyshev et al. 2017]. These can be used as evidence in investigations related to serious crimes like drug trafficking, kidnapping, murder, and often are the only source of hope solving a case. However, accessing the content of mobile phone's storage can be hindered by various factors, including encryption, undisclosed or unknown passwords and physical damage, for example.

Physical access to device's memory has been more difficult since the introduction of Full-Disk Encryption (FDE), specifically after many security improvements released in Android 4.4. This scheme encrypts the whole content of the storage memory, making methods such as chip-off and JTAG almost always ineffective [Loftus et al. 2017].

Android 10 and greater homogenized the Platform Security turning mandatory a different and rather more secure encryption scheme called File Based Encryption (FBE) which assigns an encryption key for each file on the system instead of the whole partition, making even harder to brute-force the single key used in the previous design

[Mayrhofer et al. 2021]. Forensic countermeasures include custom bootloaders (for full file system dump or password bypass, if applicable), exploitation of bugs, side-channels, studies about how a user creates a password [Markert et al. 2021] and social engineering to compile better dictionaries, among other techniques for password discovery. Once a device is unlocked, keys can be extracted from memory and artifacts can be parsed using methods similar to the one described by [Groß et al. 2021].

Unlike Apple mobile phones, which are only compatible with a limited set of known iOS versions and an even smaller set of hardware manufactured by Apple itself [Afonin and Katalov 2016], Android devices undoubtedly feature a more complex ecosystem of operating system versions and security patches updates, including a handful of makers of chipsets (e.g. Unisoc, Qualcomm, MediaTek and many others) each owning a set of semiconductor models, which make possible for some security flaws to be exploitable on a given combination of model and software configuration.

One specific component of Android Security Platform of interest is GateKeeper, responsible for implementing user's authentication in a Trusted Execution Environment (TEE) and logic for consecutive wrong password attempts, imposing a timeout as penalty and even increasing for successive errors, throttling brute-force attempts [Google 2018]. A summary of the expected timeout after n-th wrong attempt is shown in Table 1.

Table 1. GateKeeper timeout penalty algorithm. Source: [Google 2018].

Number of consecutive wrong passwords (x)	**Calculated timeout penalty (seconds)**
[0, 4]	0
5	30
[6, 10]	0
[11, 29]	30
[30, 139]	$30 * (2^{((x-30)/10)})$
[140, inf)	1 day

However this penalty system has been acknowledged to have different implementations among manufacturers [Potockỳ and Štulrajter 2022, Google 2018], differing in timeout increasing steps to even admit fixed timeouts, still providing a theoretical vulnerability for the development of automatic brute-force methods.

In light of the complex landscape described above, this article introduces a system for screen lock brute-force of Android Phones. The subsequent sections of this paper are organized as follows: Section 2 reviews relevant and related works focusing on attacks targeting password security in mobile devices. Building upon this foundational understanding, Section 3 introduces the proposed system for performing a comprehensive brute-force process, leveraging the capabilities of USB-OTG (On-the-Go) and HID (Human Interface Device). The obtained results from the system are presented in Section 4, offering insights into the efficacy and performance of the approach across various vulnerable devices. Moving on to Section 5, a thorough discussion is provided, analyzing the implications of the results within the context of digital forensics and device security. Finally, Section 6 concludes the article by summarizing key findings, discussing broader implications, and suggesting potential avenues for future research.

2. Related Works

Android system initially offered three lock options: no lock, PIN or alphanumerical (text) password. The PIN consists of a numerical password, while alphanumerical option includes characters in upper and lower case, as well as symbols, with a length ranging from 4 digits/characters to 16. This provides a large key space for brute-force attempts.

After the introduction of the Android Lock Pattern (ALP) as a screen lock alternative in 2008, which involves a 3x3 grid of contact points connected according to a set of restrictions (e.g., the number of points must be between four and nine), significant effort was devoted to probing its security. The ALP offers a trade-off between security and ease of use due to its more limited range of different passwords available for selection.

Side-channels methods for password discovery, such as analysing smudges and oily traces left on screen [Aviv et al. 2010, Cha et al. 2017] , and using captured footage from video cameras for finger path guessing, as reported in [Ye et al. 2017, Ye et al. 2018], cab be combined with social engineering, psychological factors and usability heuristics for password creation. These approaches can create a shortlist of possible candidates for pattern passwords [Andriotis et al. 2016] or PINs [Markert et al. 2021].

Despite showing promising results in theory, side-channels attacks and social engineering are not always very effective in real scenarios, specially with prevalence of more complex patterns or adoption of PIN passwords. This motivates further investigation into alternatives, such as exploitation of HID (human interface device).

2.1. ALPFinder

The work by [Gómez 2018] proposed ALPFinder (i.e. Android Lock Pattern Finder) for brute-forcing pattern lock screens. The setup consisted of an Arduino Leonardo board connected to the device with an USB-OTG cable (see Figure 1). The solution involved specific code that emulated mouse HID, simulating the swipe unlock pattern.

Proof-of-concept experiments were conducted on both real devices and software emulators, yielding satisfactory results. Although the system was straightforward and did not support PIN locks, it provided a significant breakthrough as an attack vector for pattern screen locks.

The main targets for this system were devices with fixed penalty timeout for consecutive incorrect passwords. The approach utilized a probabilistic dictionary of patterns with predefined lengths, which had to be uploaded and integrated into the code running on the Arduino Leonardo.

An important reported aspect was the maximum expected time for unlock based on the pattern's length. Even when using a dictionary attack approach, the brute force process could take weeks, and detecting a successful unlock was not feasible.

Similarly, because both the program and the pattern list were stored in Leonardo's ROM, there was no permanent logging. The entire code had to be recompiled to attempt alternative dictionaries or to accommodate delays in drawing patterns of varying lengths.

2.2. Android-PIN-Bruteforce

Another approach suggested by [Potockỳ and Štulrajter 2022] involved implementing a tethered system (shown in Figure 1) which leveraged the HID capabilities of an Android device to brute force the PIN of the target device. The tool, Android-PIN-Bruteforce, presents a Command-line interface on the host device, allowing configurations such as PIN length, timeout duration, and password dictionaries to be set before initiating the process.

An experiment was conducted using ten devices selected from different brands, models and Android versions. The test concentrated solely on the devices' ability to unlock the screen once the correct password was sent and on the repeatability of the brute-force process when an incorrect password was supplied. The study did not measure the total time required for password discovery.

Overall, compatibility was robust among devices with both micro-USB and USB type-C connections. Only one device was reported as incompatible, likely due to the Android versions bellow 5.0 lacking adequate USB-OTG support.

A crucial observation is the absence of pattern unlocking support; the system relies solely on PINs. Furthermore, the system depends on the battery life of both the target device and the host device. If the battery is depleted, the system shuts down without an option to resume the attack, necessitating a restart.

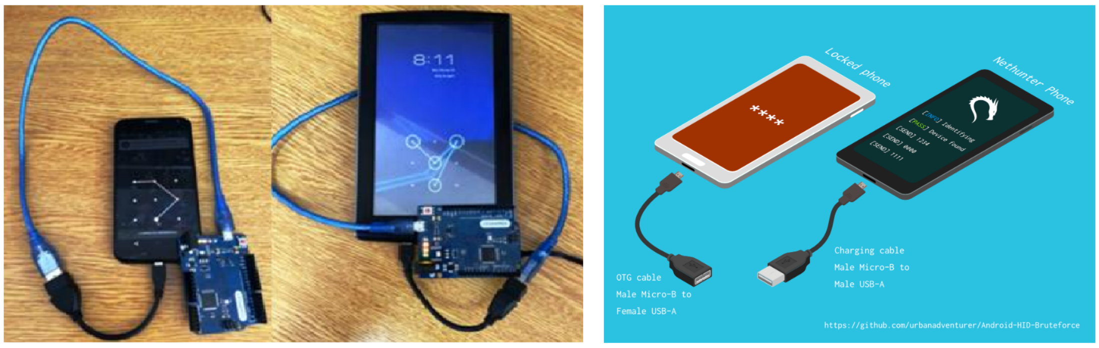


Figure 1. On the left: UI of the Controller module with main actions available for user. On the right: output of the brute force process on a real case device, highlighted PIN password discovered and unlock detection. Source: [Gómez 2018] and [Potocký and Štulrajter 2022].

3. System design

During a preliminary investigation, we noticed that often when the correct password is input on a device, its screen lights up with brighter than in a hold state. On the contrary, when the wrong password is entered, the display fades to black a few seconds after the invalid password text warning. This behaviour can be exploited for an unlock detection system.

We propose a novel system composed of two modules: a Controller application that communicates wireless with a Worker microcontroller module. The Worker module emulates mouse and/or keyboard HID accordingly to the screen lock type (pattern or PIN, respectively) and is physically connected to device being exploited by an USB-OTG cable.

The Worker module also features a light sensor for capturing the actual display brightness. It is capable of transmitting this data back to the Controller, enabling automatic detection of successful unlock attempt and triggering the brute-force process to stop.

This system allows low level device access and password input (either PIN, pattern, or even alphanumeric) to be physically isolated from a higher-level control module. This isolation enables functionalities like stop/resume, calibration, and unlocking, which can be challenging to implement on a microcontroller.

3.1. Controller module

The Controller module provides a generic and abstract application for brute-forcing passwords, implementing the main logic, and presenting a user interface for selecting password dictionaries, specifying the type of screen lock, calibrating device settings, storing the off-screen brightness value threshold (for auto unlock detection), maintaining an overall communication log with timestamps, and initiating start/resume of brute-force attacks.

Firstly a Bluetooth serial connection is established with the Worker module. Then, the user then chooses a dictionary text file and selects the type of password lock on the device, calibrates the photosensor value threshold, and enables the brute-force process to start.

Once the password is sent to the device, the Controller module waits for a predefined delay, around a few hundred milliseconds, before probing the current screen brightness. Typically, the screen lights up briefly after a successful unlock. If the algorithm detects an unlocked screen, the process halts, and the UI informs the user that the password has been discovered. If not, a timeout period is awaited as a penalty for wrong password attempts.

It's important to note that the type of lock screen must be configured because drawing a pattern on the screen takes significantly more time compared to a PIN password. Thus, the algorithm must account for this variation in waiting duration.

A calibrated value for screen brightness must be acquired before initiating the process to prevent false unlocks triggered by ambient light. See Algorithm 1 for the pseudocode of the Controller module.

3.2. Worker module

The Worker module is based on a microcontroller board with built-in USB HID capabilities, physically connected to the phone with an USB-OTG cable (either USB type-C or microUSB) and receives commands from the Controller via wireless communication. A photosensor is connected to one of the microcontroller inputs to measure display brightness.

It implements a straightforward protocol to respond actions sent from the Controller. It emulates a mouse for drawing the pattern or simulates a keyboard to input the PIN to the device. As the password is received as text, the Worker module must parse it and proceed accordingly based on the lock type. For example, upon receiving '1234', it must simulate moving the cursor while keeping the left mouse button pressed, traversing from position '1' to '2', '2' to '3', and '3' to '4', if it is a pattern password. Alternatively, if a PIN is configured, it should input '1234' in the same manner as a normal keyboard HID.

Algorithm 1 Controller module algorithm pseudocode

```
Require: $lock \leftarrow PIN$ or $Pattern$
Require: $photoThreshold$
Ensure: $connectedWithWorkerModule$
  $passwordFound \leftarrow$ False
  while ($passwordFound$ = False) and ($listPasswords >= 0$) do
      $roundPassword \leftarrow listPasswords.Next()$
      SEND $roundPassword$ to Worker module
      if $lock$ is PIN then
          $inputDelay \leftarrow 200$
      else if $lock$ is Pattern then
          $inputDelay \leftarrow 400 + (roundPassword.Length() \times 2000)$
      end if
      WAIT $inputDelay$ milliseconds
      READ $photoSensor$ value from Worker module
      if $photoSensor > photoThreshold$ then
          $passwordFound \leftarrow$ True
          PRINT $passwordFound$
      else
          WAIT wrong password timeout penalty for next attempt
      end if
  end while
```

This module also is responsible for capturing light data from the photosensor, which should be attached to the display. It sends this data back to the Controller to be used as input for the unlock detection algorithm.

Considering that devices have varying screen sizes, especially in case of a pattern lock, and different phone ”wake” keystroke combinations, it’s essential to perform a brief process of calibration. This process records the mouse pointer’s *x,y* offsets distances, both absolute and relative to other dots, accordingly to the 3x3 dots matrix (for detailed implementation, see [Gómez 2018]), as well as validate the correct sequence of keys to ”wake” a phone. Once the calibration process for a specific model is completed, the values can be saved and reused. In some cases, these values can be shared among different models or even among among users of the system.

3.3. Proof of concept

The proposed system was developed as a proof-of-concept implementation and utilized for experimentation. The prototype utilizes an Arduino Pro Micro board featuring an ATmega32u4 microcontroller. This board is equipped with micro-USB input/output, a Bluetooth serial module (HC-05) for communication with the Controller, and a GL5528 model photoresistor for light sensing. To facilitate connection, a pair of USB-OTG cables was employed: one designed for USB type-C connectors and another for the micro-USB standard.

The Worker module assembly can be observed on the right side of Figure 2 . Notably, an elastic band is used to attach the photoresistor to capture display brightness,

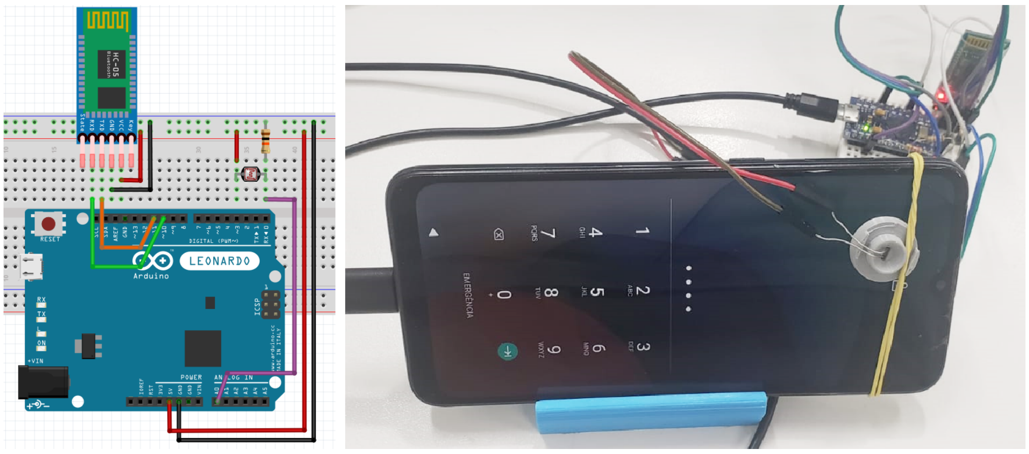
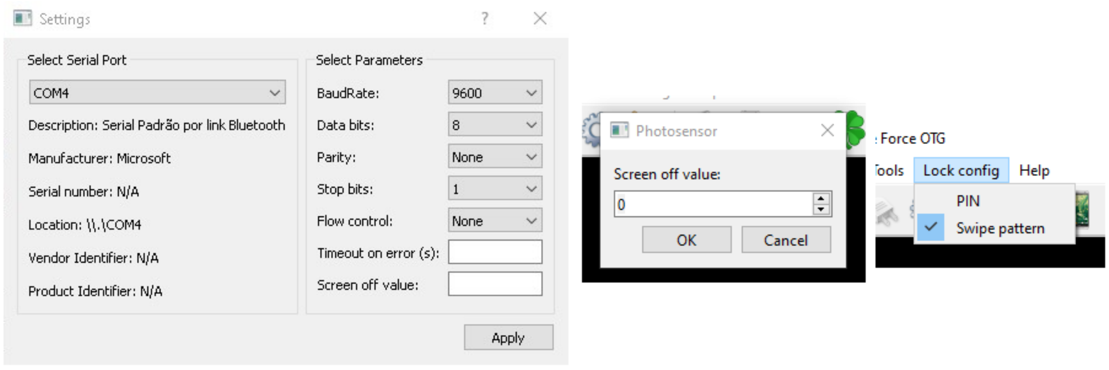


Figure 2. On the left: Schematic of the system design, presenting an Arduino Leonardo, Bluetooth module and light sensor (photo sensor). On the right: Worker module of proof-of-concept, featuring Arduino Pro Micro, Bluetooth module and photo sensor installed on the screen with an elastic band.

while the device is connected via USB type-C to the Arduino Pro Micro running the Worker module code.

Further configuration screens are depicted in Figure 3. These screens encompass setup parameters for establishing a wireless connection with the Worker, reading the photoresistor value for calibration, and selecting either a PIN or pattern lock type. These configurations are essential prerequisites before initiating the brute-force process.



Settings

Select Serial Port

COM4

Description: Serial Padrão por link Bluetooth

Manufacturer: Microsoft

Serial number: N/A

Location: \\.\COM4

Vendor Identifier: N/A

Product Identifier: N/A

Select Parameters

BaudRate:	9600
Data bits:	8
Parity:	None
Stop bits:	1
Flow control:	None
Timeout on error (s):	
Screen off value:	

Apply

Photosensor

Screen off value: 0

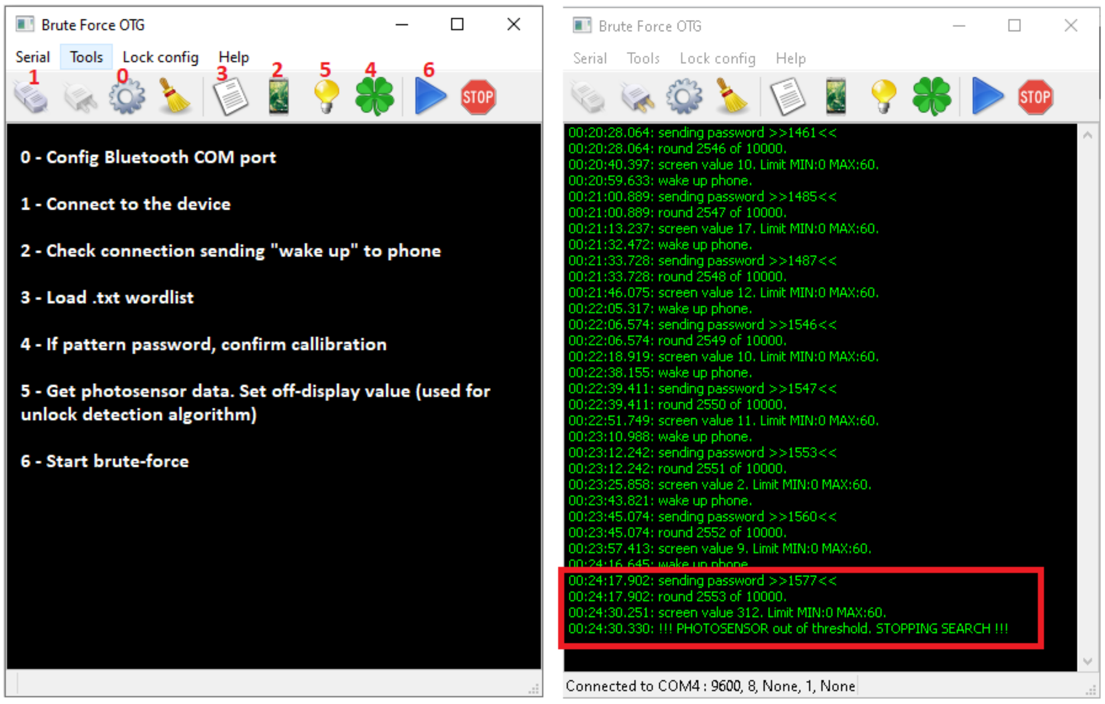
OK Cancel

Figure 3. On the left: serial Bluetooth connection setup parameters UI. Center: photo sensor calibration UI. On the right: lock screen type selection menu.

The user interface of the Controller module is presented in Figure 4. The left side of the image provides an overview of available activities. The right side demonstrated the interface after a successful attempt highlighting when the brute-force process stopped and displaying the discovered password.

4. Experimental procedure

An experiment was designed to evaluate the efficacy and effectiveness of the brute-force OTG HID method and the prototype system. It involved testing these method with compatible devices, including with a fixed timeout penalty and increasing penalty interval



0 - Config Bluetooth COM port

1 - Connect to the device

2 - Check connection sending "wake up" to phone

3 - Load .txt wordlist

4 - If pattern password, confirm callibration

5 - Get photosensor data. Set off-display value (used for unlock detection algorithm)

6 - Start brute-force

```
00:20:28.064: sending password >>1461<<
00:20:28.064: round 2546 of 10000.
00:20:40.397: screen value 10. Limit MIN:0 MAX:60.
00:20:59.633: wake up phone.
00:21:00.889: sending password >>1485<<
00:21:00.889: round 2547 of 10000.
00:21:13.237: screen value 17. Limit MIN:0 MAX:60.
00:21:32.472: wake up phone.
00:21:33.728: sending password >>1487<<
00:21:33.728: round 2548 of 10000.
00:21:46.075: screen value 12. Limit MIN:0 MAX:60.
00:22:05.317: wake up phone.
00:22:06.574: sending password >>1546<<
00:22:06.574: round 2549 of 10000.
00:22:18.919: screen value 10. Limit MIN:0 MAX:60.
00:22:38.155: wake up phone.
00:22:39.411: sending password >>1547<<
00:22:39.411: round 2550 of 10000.
00:22:51.749: screen value 11. Limit MIN:0 MAX:60.
00:23:10.988: wake up phone.
00:23:12.242: sending password >>1553<<
00:23:12.242: round 2551 of 10000.
00:23:25.858: screen value 2. Limit MIN:0 MAX:60.
00:23:43.821: wake up phone.
00:23:45.074: sending password >>1560<<
00:23:45.074: round 2552 of 10000.
00:23:57.413: screen value 9. Limit MIN:0 MAX:60.
00:24:16.645: wake up phone.
00:24:17.902: sending password >>1577<<
00:24:17.902: round 2553 of 10000.
00:24:30.251: screen value 312. Limit MIN:0 MAX:60.
00:24:30.330: !!! PHOTOSENSOR out of threshold. STOPPING SEARCH !!!
```

Connected to COM4 : 9600, 8, None, 1, None

Figure 4. On the left: UI of the controller module with main actions available for user. On the right: output of the brute-force process on a real case device, highlighted PIN password discovered and unlock detection.

algorithm for either pattern or PIN locks. The maximum attempt time per device was limited to two weeks due to feasibility. Efficacy assessment was conducted by recording the overall system performance, compatibility and identifying pitfalls.

Due to time constraints, dictionaries containing the most common PIN and patterns of a given length, as well customized lists compiled from social engineering, were used in the attacks. The total time taken for successful password discovery was recorded as well as the wordlists used and password attempt rates of the implemented system.

If the password could not be found with the assigned maximum time, it was marked ”Negative”, and a commercial forensic tool[1] was used to attempt to discover it.

If the password was successfully retrieved using the commercial forensic tool, it was added to a wordlist to validate if the brute-force OTG HID implementation could have potentially retrieved it, given sufficient time. Even if the system could unlock the device using the newly improved list, the result for that specific device remained labeled as ”Negative”.

The commercial forensic tool had limited availability due to high demand from other cases. We acknowledge its significantly greater efficiency, which, in most cases, enables the breaking of passwords in a fraction of the time required for the completion of brute-force OTG HID method. This efficiency has been recognized across-the-board by various law enforcement agencies and serves as a benchmark for evaluating the effectiveness of this work.

[1]Cellebrite UFED/Premium ES – https://www.cellebrite.com

5. Results & Discussion

A total of 18 devices were randomly selected from models known to be compatible with brute-force OTG HID method. These devices had a working display and USB connector, and were selected from cases with forensic interest, where the password was either unknown or not provided.

These mobile devices came from three manufacturers: Motorola, Samsung and Xiaomi, featuring six chipsets (different names for commercial models may share the same chipset[2]). The devices were launched between 2013 and 2021, and they ran Android versions ranging from 5 up to 10.

The devices were also categorized into three groups based on the timeout penalty algorithm. Group A included devices with a 30-second penalty after the 5th incorrect password attempt. Group B had a 30-second penalty after every incorrect attempt. Group C devices had an incremental penalty.

As detailed in Table 2, results indicated that 66% of the devices were compatible with the brute-force OTG HID technique with a positive password discovery. There was considerable variability in the duration of password brute-force, ranging from one hour to up to eight days, with an average of more than one day for successful unlocks.

Table 2. List of examined devices.

Manufacturer	Chipset	Timeout group	Lock type	Brute-force OTG	Time	Forensic Tool
Motorola	MT6762G	B	PIN	Positive	3 days	
Motorola	MT6762	B	PIN	Positive	16 hours	
Samsung	MT6765	C	PIN	Negative		Positive
Samsung	MSM8916	A	PIN	Positive	8 days	
Motorola	MT6762G	B	PIN	Positive	3 days	
Motorola	MT6762G	B	Text	Negative		Negative
Motorola	MT6762G	B	Pattern	Positive	1 day	
Samsung	SC9863A	C	PIN	Positive	1 hour	
Motorola	MT6762G	B	PIN	Positive	6 hours	
Motorola	MT6762G	B	PIN	Positive	1 hour	
Motorola	MT6762	B	PIN	Negative		Positive
Samsung	MT6739	C	Pattern	Negative		Negative
Motorola	MT6762	B	Pattern	Negative		Positive
Motorola	MT6762	B	PIN	Positive	1 hour	
Motorola	MT6762	B	PIN	Positive	5 day	
Motorola	MT6762G	B	PIN	Positive	4 day	
Motorola	MSM8226	A	Pattern	Positive	1 day	
Xiaomi	SDM439	C	Pattern	Negative		Positive

Only one device examined was locked with alphanumerical (text) password that could not be retrieved by either tool. Pattern passwords were discovered in only 2 out of 5 devices (40%). However, the proposed system successfully discovered the passwords

[2]Specific models or commercial names withheld for ethical and security reasons.

of most devices locked with PINs (10 out of 12), were compatible with the brute-force process (see Figure 5).

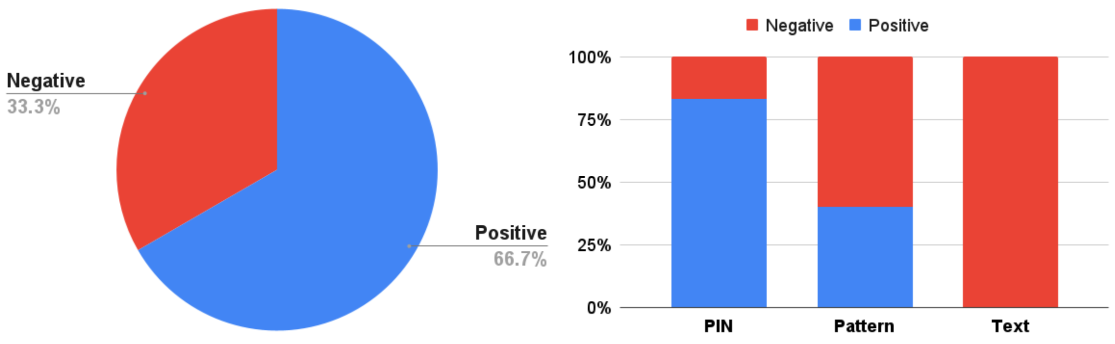


Figure 5. Left: Brute-force OTG HID results showed that 66.7% of the devices had a positive outcome, while 33.3% had a negative. Right: Among devices locked with a PIN, 89% could be cracked, and 40% of those with a pattern. No alphanumerical (text) passwords could be cracked within the designated time.

All devices in group A, which used a less secure timeout algorithm, and nine out of twelve (75%) devices classified in group B, could be brute-forced. The PIN password was only discovered on one of the four devices that had incremental timeout penalties (group C), as shown in Figure 6.

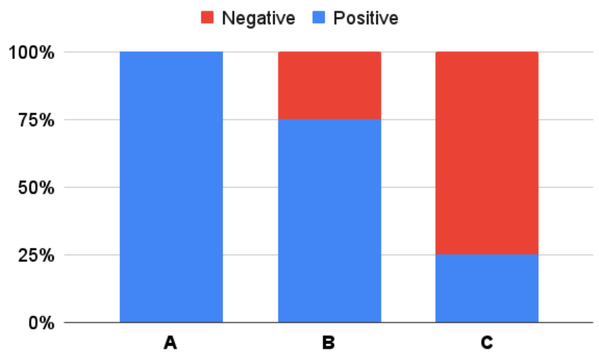


Figure 6. Brute-force results by timeout penalty algorithm group. Group A (less secure penalty) had password retrieved from all the devices. Group B (medium security): 75% of the devices were compatible with password discovery. Group C (incremental penalty): one out of four (25%) were positive.

All passwords retrieved by the forensic tool could be confirmed to work with brute-force OTG HID system, and were subsequently added to a wordlist of known patterns and PINs. Even complex swipe patterns consisting of lengths eight and nine could be correctly emulated by HID and accounted for using the proposed delay computation algorithm. Out of the devices with negative results with the OTG HID brute-force method, only two could not be discovered by the forensic tool.

Lastly, the recorded attempts per hour (APH) for both PIN and pattern screen locks, for Group A and Group B timeout algorithms are presented in Table 3. The data shows no variation in rates for PIN passwords despite their length, providing a more precise anticipation of the maximum brute-force duration. In contrast, rates for pattern locks are notably lower and change in accordance with their length. Longer patterns take significantly more time to exhaust the dictionary compared to smaller ones. For instance, pattern length 6, despite being only 2.6 times the key space of PIN length 4, can take up to approximately 10 times longer to exhaust the dictionary.

Table 3. Group A and Group B devices fixed timeout penalty algorithms and maximum expected time to exhaust the dictionary.

Timeout penalty algorithm	**Attempts per hour (PIN)**	**PIN (length 4)**	**PIN (length 6)**	**Pattern (length 4)**	**Pattern (length 6)**
Group A (30s penalty after the 5th incorrect)	360	28 hours	4 months	15 hours (APH 110)	12 days (APH 92)
Group B (30s penalty after each attempt)	100	5 days	14 months	24 hours (APH 70)	18 days (APH 62)

5.1. Discussion

In total, twelve out of eighteen devices (66%) could be brute-forced within a reasonable short period of two weeks. However, it must be noted that only a fraction of the total possible patterns, for example, was tested, underscoring the crucial need for a concise, probabilistic, and reliable dictionary.

Regarding PIN locks, their length is variable between 4 to 16 digits, theoretically offering greater security due to the significantly larger key space. Nonetheless, the results revealed that most PINs are actually much shorter, and effective dictionaries can be generated using social engineering information or probabilistic data from prior successful attempts. Paradoxically, despite the smaller key space, pattern passwords proved to be more difficult to retrieve than PINs, suggesting that alternatives to HID should be explored.

Despite the limited sample size of only a single device configured with an alphanumeric (text) password, brute-forcing it was significantly more challenging. This was due to the fact that the password could comprise letters, numbers, and special characters, adding to the immense complexity of the key space. The vastness of the key space made the use of the proposed system nearly impractical, even when considering only a fraction of the available key combinations. In fact, if we take into account the scenario of increasing penalty throttling, alphanumeric passwords could impose significant difficulties even for the commercial forensic tools, considering the time constraints.

While one device was not compatible with the commercial forensic tool during the experiments, it remained compatible with HID emulation and thus with the brute-force OTG HID method. This suggest a potential application of this system as a final alternative for devices that are incompatible with forensic tools.

During longer sessions, it was observed that the device's battery sometimes ran out. However, in all cases, the brute-force process could be resumed using the log of the Controller module. An alternative for extended sessions is to use a special power connector directly attached to the main logic board of the device or to employ an external power source, replacing device's battery with a continuous source.

False detection of unlocks was relatively common in devices with clock alarms configured. Given this, a false-positive detection is much preferable to missing a true unlock, as the process can always be resumed. However, once a password has been verified, it should not be retested to avoid wasting time. Furthermore, false detection was rarely observed in pattern-locked devices but more common in PIN-locked devices.

6. Conclusion

In this work we proposed a novel system consisting of two modules for attacking lock screen protections on Android devices using HID brute-force via USB-OTG technique.

Our system identifies a correct password attempt by detecting changes in screen brightness, exploiting the increase in light that devices emit after a unlocking. This approach is particularly valuable, as the extended wait times required for brute-force attempts, ranging from days to weeks, render a manual unlock detection method unfeasible.

We designed an experiment to assess the efficacy and effectiveness of the proposed system using a prototype implementation, and we used a commercial forensic tool as an alternative benchmark for password discovery.

The results revealed that the correct password was successfully retrieved from approximately 66% of the devices within a span of up to two weeks. This outcome validates the system as a viable option, especially for devices with less secure implementations of timeout penalty algorithms. Moreover, considering the overall low cost of building the apparatus and the minimal risk to the examined evidence, the system presents a promising solution for device triage as well.

A potential future approach to mitigate throttling enforcement by the GateKeeper Android component involves exploiting vulnerabilities in the USB-OTG protocol or resetting the failed password counter after each attempt. This strategy aims to avoid disrupting password authentication and could allow for higher attempt rates, facilitating a more comprehensive search across the entire key space.

Acknowledgment

We would like to thank all forensic experts, forensic assistants and personnel from Polícia Científica de Santa Catarina for their valuable support and contribution to this research project. Implemented code is available at `https://github.com/matnunes/BruteForceOTG`.

References

Afonin, O. and Katalov, V. (2016). *Mobile Forensics – Advanced Investigative Strategies*. Packt Publishing Ltd.

Andriotis, P., Oikonomou, G., Mylonas, A., and Tryfonas, T. (2016). A study on usability and security features of the android pattern lock screen. *Information & Computer Security*, 24(1):53–72.

Aviv, A. J., Gibson, K. L., Mossop, E., Blaze, M., and Smith, J. M. (2010). Smudge attacks on smartphone touch screens. *Woot*, 10:1–7.

Cha, S., Kwag, S., Kim, H., and Huh, J. H. (2017). Boosting the guessing attack performance on android lock patterns with smudge attacks. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 313–326.

Chernyshev, M., Zeadally, S., Baig, Z., and Woodward, A. (2017). Mobile forensics: advances, challenges, and research opportunities. *IEEE Security & Privacy*, 15(6):42–51.

Gómez, L. S. M. (2018). Descubrimiento automatizado de patrones de acceso en dispositivos móviles android. In *XVIII Simposio Argentino de Informática y Derecho (SID)-JAIIO 47 (CABA, 2018).*

Google (2018). Android open source project: Android api guide.

Groß, T., Busch, M., and Müller, T. (2021). One key to rule them all: Recovering the master key from ram to break android's file-based encryption. *Forensic Science International: Digital Investigation*, 36:301113.

Loftus, R., Baumann, M., van Galen, R., and de Vries, R. (2017). Android 7 file based encryption and the attacks against it. *University of Amsterdam*, 33.

Markert, P., Bailey, D. V., Golla, M., Dürmuth, M., and Aviv, A. J. (2021). On the security of smartphone unlock pins. *ACM Transactions on Privacy and Security (TOPS)*, 24(4):1–36.

Mayrhofer, R., Stoep, J. V., Brubaker, C., and Kralevich, N. (2021). The android platform security model. *ACM Transactions on Privacy and Security (TOPS)*, 24(3):1–35.

Potockỳ, S. and Štulrajter, J. (2022). The human interface device (hid) attack on android lock screen non-biometric protections and its computational complexity. *Science & Military Journal*, 17(1):29–36.

Reedy, P. (2020). Interpol review of digital evidence 2016-2019. *Forensic Science International: Synergy*, 2:489–520.

Ye, G., Tang, Z., Fang, D., Chen, X., Kim, K. I., Taylor, B., and Wang, Z. (2017). Cracking android pattern lock in five attempts. In *Proceedings of the 2017 Network and Distributed System Security Symposium 2017 (NDSS 17)*. Internet Society.

Ye, G., Tang, Z., Fang, D., Chen, X., Wolff, W., Aviv, A. J., and Wang, Z. (2018). A video-based attack for android pattern lock. *ACM Transactions on Privacy and Security (TOPS)*, 21(4):1–31.