Earworms Make Bad Passwords: An Analysis of the Nokē Smart Lock Manual Override

Jack McBride, Julio Hernandez-Castro, Budi Arief
School of Computing, University of Kent, United Kingdom
Email: {jlgm2, jch27, B.Arief}@kent.ac.uk

***Abstract*—This paper presents a security analysis of the manual override feature of the Nokē smart lock. The Nokē allows its user to operate, monitor and even share his smart lock with others through a smartphone. To counter the risk of being unable to open the lock when the smartphone is unavailable, it provides an override mechanism. Nokē implements this override feature using a quick-click scheme, whereby its user can choose a sequence of eight to sixteen short and long shackle presses (similar to a Morse code). To explore the security implications of this feature, we conducted a study collecting human-generated quick-click codes from 100 participants, and analysed and modelled the resulting dataset. Our analysis shows that the override mechanism, at least in its current implementation, presents a significant opportunity for successful guessing attacks. We demonstrate this by building a mechanical brute force tool that on average can test one quick-click code in under three seconds. We conclude that this speed, together with the low entropy of human-generated passcodes, makes this manual override feature one of the most significant weaknesses of the system and constitutes a promising attack vector. We responsibly disclosed our findings to the Nokē manufacturer. We also provide a list of potential countermeasures that can help to address this risk. We believe that alternative authentication methods such as quick-click codes will become increasingly popular in ever-expanding Internet of Things devices, so the weaknesses and the countermeasures discussed in this paper are timely and relevant, as they can also apply to other devices and security systems that rely on unconventional user-generated authentication codes.**

***Keywords*-security; brute force attack; smart locks; Internet of Things; user study; passcode selection; override mechanism.**

I INTRODUCTION

The Internet of Things (IoT) has the potential to make lives more comfortable and effortless, through various assistive products and services built using small, wireless devices; for example, to enable personalised services (in which the user gets their environment configured and presented to their preference) or multi-factor effortless and continuous authentication (where the user does not need to remember burdensome passwords but uses instead biometrics and other wearable tokens). However, these devices could also pose new large-scale privacy and security risks that are not properly understood yet, and constitute an ongoing research challenge.

The recent Distributed Denial of Service (DDoS) attacks utilising a massive botnet of compromised IoT devices [1][2] highlights the serious risks posed by insecure IoT systems further.

One of the biggest issues we face in this area is the rapid growth of off-the-shelf IoT devices available on the market. These can be quickly configured, connected and operated even by users who have limited familiarity with security. Available products range from smart versions of domestic appliances (such as light-bulbs and coffee machines [3]) to home automation systems (especially home security and environment control kits [4]) and connected vehicles [5]. Several research papers have recently covered the vulnerabilities that commonly plague these devices, some of which are built upon commercial microcontrollers such as the Arduino platform [6]. The impact of these vulnerabilities has been clearly demonstrated; recent work has documented the weaponisation of hundreds of devices into IoT botnets [7], as well as the potential deployment of chained attacks capable of disrupting large-scale networks featured in smart cities [8].

In this paper, we focus our investigation on smart (IoT) locks, which are growing in prevalence due to their versatility, from securing doors, lockers and sheds to bikes and other portable items. Our research aims to contribute to the investigation of the way user-defined security is implemented in current smart locks available on the market, by analysing the strengths and weaknesses of user-based heuristics, as well as through exploring the rationale behind the choices made by users. Furthermore, we want to examine in detail the impact of the different trade-offs between usability and security in smart locks, and extract lessons that can contribute to future, more robust and user-friendly IoT devices.

The rest of the paper is organised as follows. Section II introduces the concept of smart locks, along with a discussion on the usability and security trade-offs related to smart locks. Section III provides an overview of the methodology used in our research. We conducted a data collection phase where participants chose and used quick-click codes and reported their rationale for the passcode and other relevant data. Section IV describes a mechanical tool that we designed and built to test our theory that it is practical and feasible to break most of these user-chosen codes within a reasonable time. Section V outlines the findings gathered from the analysis of our data collection, which led to the creation of a simple Markov model to reflect the human biases previously observed in these codes. Section VI discusses the impact of our results and highlights the vulnerability caused by the override mechanism, as well as potential countermeasures. Section VII concludes our paper and Section VIII outlines ideas for future work.

II Smart locks and their usability and security trade-offs

There are several popular smart locks available on the market, notably the Danalock[1], Masterlock[2] and Nokē[3]. The smart lock market is expected to reach a value of $3.6 billion by 2019[4]. Such locks can also be used in bike-sharing schemes [9], which will expand their spread further.

One of the key requirements of a smart lock system is to have the owner pair up their smartphone with their smart lock so that the two can interact. This allows the owner to control and monitor the lock remotely, or even to share it with other users.

In contrast to their classical counterparts, smart locks usually provide an override method to unlock them in case the preferred method (smartphone control) is not available. This is useful, for example, when the smartphone is stolen, lost or otherwise unavailable.

Similar override features can be found in many commercial non-IoT security products, such as TSA-approved padlocks, which is commonly used by travellers coming to the USA. Such padlocks allow TSA staff to inspect and re-lock luggage without the need to break the padlock. For this to be possible, TSA personnel use a master key [10] to override the padlock's security mechanism (which is typically based on a physical key or a numerical combination set by the owner) using the TSA master key. However, while the intention here is to assist TSA staff and protect consumer padlocks, this override feature broadens the attack surface considerably. An attacker can easily use basic lock-picking techniques to bypass the master key's point of entry, hence significantly reducing the time needed to open the padlock, while at the same time, concealing any evidence of tampering [11]. Consequently, this override mechanism significantly decreases the security of the whole system and frequently constitutes the preferred and easiest way to bypass the security measures put in place.

In smart locks, manual override is a popular and arguably necessary feature. This is due to the real risk that the smartphone may become unavailable due to a drained battery, theft or loss. However, the way this manual override is implemented varies greatly from one smart lock to the next, as there are no standards or best practices that can aid smart lock designers. For example, the Masterlock implements an override known as the "primary code", which is a user-defined sequence of 7 button presses on the device's directional pad (four possible options: up, down, left and right), which – when entered correctly – will unlock the device [12][13]. The code is initially set up via the Masterlock application on the smartphone, but upon being saved to the device, it will function independently of the user's phone or tablet. In comparison, the Nokē smart lock uses a quick-click code [14]. This is a sequence of Morse-like clicks, which are set and operated by pressing the shackle of the device in a sequence of short and long presses. During the initial setup of the Nokē, the application on the smartphone offers the user a choice between opting for a randomly generated quick-click code of sixteen characters or defining their user-generated code with a length ranging from eight to sixteen characters.

The lack of guidelines or any other suitable heuristic makes not only the implementations of this override feature vary widely, but also the user response to them highly inconsistent. This presents a unique challenge and opens up a possibility of design flaws and further weaknesses in the system's security. This is made worse by the fact that – as we found out during our investigation – it is not possible to disable the quick-click code feature. Furthermore, the Nokē lock does not appear to log any records reflecting neither failed nor successful attempts to unlock it by using the quick-click code.

The role of the mobile device (smartphone or tablet) as an "access token" for smart locks presents an interesting issue. By encouraging users to rely on the mobile device for trivial tasks, such as unlocking their front door or accessing their valuables, successful authentication becomes heavily skewed towards the requirement of possessing the device. In this case, multi-factor authentication is achieved through two means: the mobile device paired to the smart lock (something the user owns) and the password credentials for accessing the application (something the user knows) [15].

The option to set a user-defined password improves human interaction by actively engaging the user in the security of a product, and enhances the memorability of the passwords [16][17][18]. The cost, however, is the associated potential for weak, exploitable passwords being picked and, as a result, the security of the system might be severely compromised. In the case of the Masterlock, a fixed primary code of length seven imposes a severe constraint on the code's entropy, turning it into an appealing attack surface for adversaries. There are cases of security systems which attempt to mitigate this problem, for example by favouring the use of a fixed, pre-defined passcode or phrase as a backup, or by allowing only the use of a randomly generated code created by the system. These alternatives to user-defined heuristics tend to come short regarding memorability, as their lack of

[1] https://danalock.com/?page_id=35
[2] http://www.masterlock.com/personal-use/product/4400d
[3] https://noke.com/pages/padlock
[4] https://goo.gl/4aFjZ1

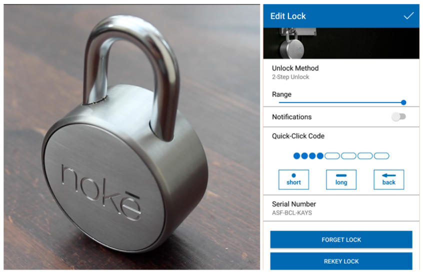

Figure 1: The Nokē smart lock (left) and the quick-click code selection screen in the Nokē application (right).

user involvement and increased complexity make recalling them a complex task for average users [19].

In this paper, we focus on the Nokē smart lock (see the left image of Figure 1), which is one of the most popular smart locks currently available on the market. Furthermore, the Nokē lock is one of the only three smart locks that have been highlighted as unbreakable in a presentation at Defcon'16[5], which makes their study more challenging and exciting. In particular, Nokē's quick-click code seems to constitute an appealing attack vector because this mechanism can be user-generated and hence is potentially open to brute force attacks.

The attack surface of the quick-click system is dependent upon several factors. First, the strength of the quick-click code is significantly affected by the users heuristic in choosing the length and the pattern of the code. Second, as with many user-defined heuristics, the inclusion of personal details encoded in the quick-click code can be subject to some forms of social engineering, particularly if those details are easily gathered. Third, as the quick-click code is physically entered in a public setting where the lock may be in use, it is possible to "shoulder surf" while the user is entering the code, even from a long distance with not too specialised equipment (binoculars, telescope). From this, critical information may be obtained regarding the structure of the code, if not the full code itself. Finally, by collecting a large enough set of quick-click codes, information about the statistical properties and other quantitative and qualitative data can be obtained, analysed and employed in an improved attack.

III Methodology

We recruited 100 participants to collect quick-click codes for later analysis. Participants were individually recruited on a voluntary basis in a public setting, where they were briefed on the study and invited to join in exchange for Amazon gift cards with a value of £10. To provide a common structure and background and to reduce or at least control *priming*, an instructional video[6] and short script was prepared and shown to all participants. The video informed them of the features of the Nokē padlock, with a focus on the quick-click code. This assisted in demonstrating the outline of the study, briefing the participant and establishing a consistent approach to their interaction.

While watching the video, participants were requested to independently create their quick-click code and save it to the smartphone provided by the researcher. Whilst this taking place, their interactions provided a source of qualitative data, in which factors such as body language and verbalisations were observed and recorded. This complemented the collection of raw data on code lengths and other characteristics such as code patterns. This data gathering phase was followed by a more advanced mathematical analysis of the properties of the quick-click codes chosen, regarding uniqueness and entropy.

The data from the first 50 participants constitute our *training data set*. This was used as a base for developing a model of human-generated passcodes. We collected further 50 participants' codes, and this new, previously unseen collected code was used as a *testing data set* for estimating the success rate of our proposed attack.

We are aware of the relevance of questions regarding the ecological validity of password studies [20], and we strive to meet this key requirement in our study.

We gathered a sample of 100 participants (74 male, 26 female) of varying ages, backgrounds and occupations and conducted semi-structured interviews with them, during which time we introduced them to the Nokē device. The interviews typically lasted between 5 and 10 minutes and involved a briefing of the study using an instructional video shown to the participant. This short video explained the functionality of the quick-click code, and provided a visual guide to show participants how to select a new code, without *priming* them to pick any particular pattern or length. The video also prompted the participant to consider their quick-click code, and to make comments on the experience. The video concluded by instructing the participant to save their code in the Nokē smartphone app.

The image on the right of Figure 1 shows the app screen for entering new codes. Once the code was successfully saved to the Nokē, the participant was prompted to unlock the device with their newly created code. During this stage, participants were made to recall their code from memory. The number of attempts required to unlock the device was observed and recorded, and a follow-up unlock attempt was requested to ensure adequate memorisation of the code. This was also used to test the feasibility of a given code.

Once the Nokē was successfully unlocked for the second time, the participant was asked "What was the rationale behind your quick-click code?", and the response was recorded. The aim here was to determine the specific reasoning behind the structure of the code, for a later comparative analysis. The participant's code and length were recorded as quantitative data.

[5]https://goo.gl/ZkT7bY
[6]https://www.youtube.com/watch?v=EbANkK3nhmA

In general, there are two groups of data collected: *quick-click data* and *observational data*.

III-1 Quick-click data: This data is on the quantitative measures of the passcodes and their statistical modelling and other properties.

The data was encoded as a series of dots and dashes to represent the sequence of short and long presses respectively (reflecting on how it was displayed on the user interface provided by the Nokē app). Gathering and studying this data was one of the primary aims of the study, and it provided a good insight on the passcode-using habits of the sample participants and potentially even larger groups. We maximised the ecological validity of this study by collecting data from a general population of smartphone users as our sample, across which there was a range of educational backgrounds, ages, and occupations. Additionally, the interviews were conducted in a public setting, where the participants were not held to any particular time constraints when choosing their codes. The Nokē smartphone app was used by participants during this selection process, to emulate the process of a real world owner selecting a code for their device. There are three sets of data collected here:

- Quick-click code: we encoded each participants quick-click code into a Morse-like sequence of dots and dashes (e.g. ● ● ● ● - - - - for a code of length 8 formed of four consecutive dots and dashes).
- Length: to allow for average code lengths to be calculated and statistical analysis to be performed on them.
- Number of attempts: to determine the feasibility of the code set by each participant.

III-2 Observational data: This was collected to form a record of both qualitative and quantitative data. Our data was split into two distinct areas:

- Information on the participants and their codes, thus encapsulating the numerical codes and their rationale. In Section V, we discuss how this data was processed to reveal patterns and tendencies.
- Comments: Words or phrases used by the participant whilst interacting with the lock and creating their code.

IV Designing and building the brute forcer

In order to demonstrate the feasibility of carrying out a brute force attack on the Nokē lock, we designed and developed a mechanical prototype that can automate and speed-up the process of entering quick-click codes (see Figure 2). The main components of the brute forcer are:

- an analogue feedback servo with an arm attached (which is strong enough to press down the shackle)
- an Arduino Uno microcontroller (for controlling the servo so that the arm can be programmed to press the Nokē shackle in a systematic way)
- a custom made 3D-printed casing (which can hold the Nokē lock and the servo in place while the latter carries out a systematic sequence of presses of the shackle)

The programming logic for the microcontroller is pretty simple. Based on our attack model, a series of potential quick-click codes are queued to be tested on the Nokē. Each of these is represented as a sequence of binary numbers (where 0 represents a short press and 1 a long press). The Arduino controller sends these codes as instructions to the servo, which then executes them as presses on the shackle. This sequence of instructions is carried out in a loop until one of two events occur: either the brute forcer finds the correct quick-click code, or it enters a "locked out" mode after 64 shackle presses have been made. This is due to Nokē's security mechanism for slowing down brute force attacks, in which Nokē will be locked out for one minute (i.e. it will not respond to any quick-click code) after 64 unsuccessful presses of the shackle. This measure reduces the performance of the brute forcer somewhat, but such an attack is still feasible and can be performed within a reasonable time, as we discuss in Section V.

To allow others to replicate our set up and results we share the blue print of the 3D casing at *https://github.com/exampleprogrammingbugs/bruteforcer*.

V Results

In this section, we present the main results of our study, from the initial phases of data analysis to an evaluation of the overall success of our approach. We describe in detail how we created a simple Markov model based on the data gathered from 50 of the 100 participants. Then, we used this model in conjunction with the brute forcer device described in Section IV to launch an attack against 50 passcodes collected from a different set of participants, to evaluate the feasibility of our brute force approach.

The full dataset of quick-click codes from 100 participants can be found at: https://goo.gl/4eCEjk. The scanned version of the questionnaire completed by the participants can be found at: https://goo.gl/4T05Se.

A set of ten samples of the quick-click codes data is presented in Table I. Each row contains data collected from one participant, including their chosen quick-click code, its length, the participant's rationale for choosing that code, some remarks from the researcher based on the observation of the participant's interaction when choosing the code, and the category that the code belongs to (one of the five categories defined in Section VI).

V-A Participant data and code length

Following the data set collection phase, we began the analysis of patterns in the passcodes and studied which factors could affect different security-related features. We sorted our data based on specific participant criteria such as age, gender and education to explore any relations between individual participant factors and the properties of their passcodes. No significant relations were found, apart from a tenuous positive correlation between the participant's age

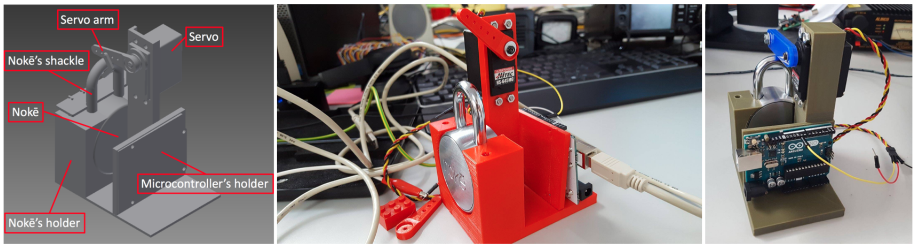

Figure 2: The 3D model and the 3D-printed prototypes of the Nokē brute forcer.

Table I: Ten samples of participant's data.

Participant	Quick-click Code	Length	Comments	Researcher Comments	Category
P3	● ● ● ● ● - - ● ● ● ● ● - - ● ●	16	"I need to think of a song...", "All the beats would be short presses"	Code inspired by Johann Strauss "The Blue Danube Waltz". The participant hummed the melody when unlocking the Nokē.	Earworm
P5	● ● - ● ● ● ● ● -	9	"Based off of a song"	Code inspired by Guns N Roses Paradise City. When creating his combination, this participant wrote it down on a piece of paper.	Earworm
P9	- ● ● ● ● ● - ● ● ●	10	"It's Bob in Morse code"	The participants code was based off of Morse code, noting the similarities between that and the Quick-Click code. It was the name Bob in Morse. They explained it was something he would have to look up online to remember how to encode.	Morse
P13	● - - - ● ● ● ●	8	"It's Jess in Morse code"	The participant designed their code after recognising the similarity between the Quick-Click code and Morse code. To make it memorable, they encoded the name Jess as their combination.	Morse
P15	● ● ● - ● ● ● - ● ● ●	11	"My favourite number is 3", "...patterns of 3 broken up by the longer dash"	This participant created a pattern based upon a repetition of their favourite number.	Numerical
P24	● ● ● ● ● - ● ● ●	9	"I just set my code and tried to remember what I chose"	The participant chose a code supposedly on a random basis - keeping it short at a length of 9, and splitting it into segments with a long press in the 6th position.	Pseudo-random
P28	- - ● - ● - - - - ● - ● - ● ● ●	16	"I memorised the long presses as a pin code", "I randomised the way the pattern was generated", "I can memorise this as 2-1-4-1-1' "	This participant tossed a coin for each press of the code - the two sides corresponding to a long or a short press. In this way, the code was pseudo-randomised. The participant then memorised the pattern of consecutive long presses as separated by the short presses, making it easier to remember as a number code rather than a sequence of presses. They rationalised that this would be easier to recall for long term use.	Numerical
P31	- - ● - ● ● - - - ● ● - ● - -	15	"I wanted to make it long and complex but with some structure"	The participant devised a complex code which featured a mirror of two 7-length sequences separated by a long press.	Pseudo-random
P55	● - ● ● - ● - ● -	9	"Used my full name and syllables in it. So last syllable in each name is a long one, rest are short"	The participant encoded their first and surname syllables by alternating between short and long presses.	Name Substitution
P85	- ● - ● ● ● - ● - ● -	11	"Easy to remember, Encoding of first and last names vowels and consonants with alternate presses"	The participant encoded their full name, using short presses for vowels and long presses for consonants	Name Substitution

and the length of their password. Included in the gathered data were direct quotes from participants while taking part in the study. This intended to capture user thoughts and behaviour when interacting with the device, as well as for the use of their codes. To document our observations, we recorded the data throughout the interview verbatim.

Figure 3 shows the breakdown of code length choices made by participants. This provided a basis for our inquiry into the way users perceived and dealt with the constraints of the code regarding length and character selection.

We processed the data from the interview stage into graphs and charts for easing visualisation, as shown in Figures 3 and 4. From this, we drew further conclusions

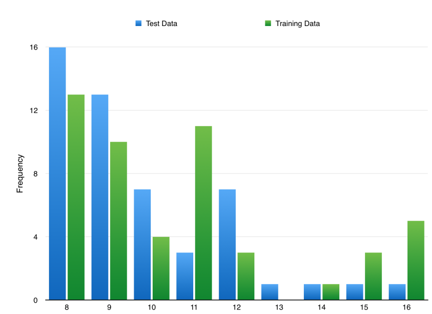

Figure 3: Distribution of passcode lengths across the training and test datasets.

about the characteristics of the codes the participants created. This analysis intended to gain insights into code patterns, as well as exploring just how unique each quick-click code is, in comparison to other quick-click codes. Whilst the sample size was relatively limited, the main characteristic we focused on was code length.

V-B Auto-generated codes

A further investigation into the automatically generated (random) codes created by the Nokē app itself was also conducted. We aimed to assess the security of these seemingly random codes by comparing them with the ones generated by the participants. By investigating this human factor, we were able to infer several statistical patterns and biases common to the human generated codes that would later be used to prepare an accurate model for the attack. Studies indicate that optimal security can only be attained with passwords that are randomly-generated [21]. As such, this became an important metric in our study; to measure the security of the randomly generated passcodes offered to users in place of defining their own.

As it is apparent in Figure 4, the randomly generated codes (right) significantly differ from the human generated ones (left). Furthermore, the resultant *random* code seems not to adhere to any specific pattern that might be advantageously exploited by attackers. It is obvious that trying to crack human generated codes will be significantly easier, particularly when the attacker knows sensitive, encodable information about the target victim – e.g. date of birth, favourite number, or a musical melody connected to the victim.

V-C Analysis of the passcodes

To complement our analysis of the participant data, we analysed the 50 codes to learn about their exploitable properties. This included looking into code likeness and uniqueness (for which we used string comparison with the Levenshtein distance), and developing a basic Markov model of the codes and studying any significant patterns.

To explore passcode similarities, we encoded passcodes into a binary form: short presses (•) were replaced with 0, and long presses (-) with 1. For example, a quick-click code of (• • • • - - - -) was encoded as 00001111.

The following sections detail how the Levenshtein distance metric and the basic Markov model were used.

V-C1 Levenshtein distance: The Levenshtein distance was used to measure the level of code uniqueness on a broader scale, by comparing codes of different lengths.

The key difference in length variation allowed for a comparison to be drawn between the participants' codes and the auto generated Nokē codes, as well as the breadth of uniqueness of participant selected codes across various lengths. A pairwise comparison of Levenshtein distances for

Table II: Chi-square test for user character distribution.

Character	Expected	Observed
0	265	304
1	265	226
χ^2	0.455	11.479

the human generated and automatically generated codes can be found in Figure 4.

Both figures have been colour-coded to emphasise patterns in the data, based on the level of dissimilarity between two codes. In this context, dissimilarity is the number of differences between two strings [22].

- Teal: Dissimilarity between 12 and 16. The two cross-referenced codes are considered uniquely different.
- Green: Dissimilarity between 8 and 12. The two cross-referenced codes have a high difference level.
- Yellow: Dissimilarity between 4 and 8. A medium distance between the cross-referenced codes.
- Orange: Dissimilarity between 0 and 4. Codes in this category have significant similarities.
- Red: Dissimilarity equal to 0. Codes in this category are identical.

V-C2 Markov model of participants' quick-click codes: One important observation is that there exists a clear user preference towards codes with more zeroes than ones. For example, on the 50 quick-click codes in the training dataset, we found 304 zeroes and 226 ones out of a total of 530 characters (for an average length of 10.6 characters per code). We computed the Chi-square statistic

$$\tilde{\chi}^2 = \sum_{k=1}^{n} \frac{(O_k - E_k)^2}{E_k}$$

corresponding (in this case) to two bins and thus 1 degree of freedom, to measure whether this was enough evidence of a strong bias towards zero in human-generated codes. The value corresponding to a p-value of 0.001 is, for that distribution, 10.827 and as the value obtained (see Table II) is $11.479 > 10.827$, we can conclude that users do indeed significantly prefer zeroes over ones in quick-click codes, with a p-value of $p < 0.001$.

Just for comparison's sake, a similar analysis of the testing data leads to the same conclusion, this time with an ever more extreme value $\chi^2 = 20.41$.

We can conclude that it seems apparent that humans prefer short shackle presses (represented by zeroes) than long presses (ones). This is an important finding for our later attack, that will somewhat improve its speed because it is significantly quicker to enter a zero (150 milliseconds) than a one (455 milliseconds) with our brute forcer device.

We continued our analysis by creating a simple Markov model based on the user generated passcodes, a model that we would use in the later study to generate likely codes to feed the brute forcer in its attack of the Nokē. For that, we

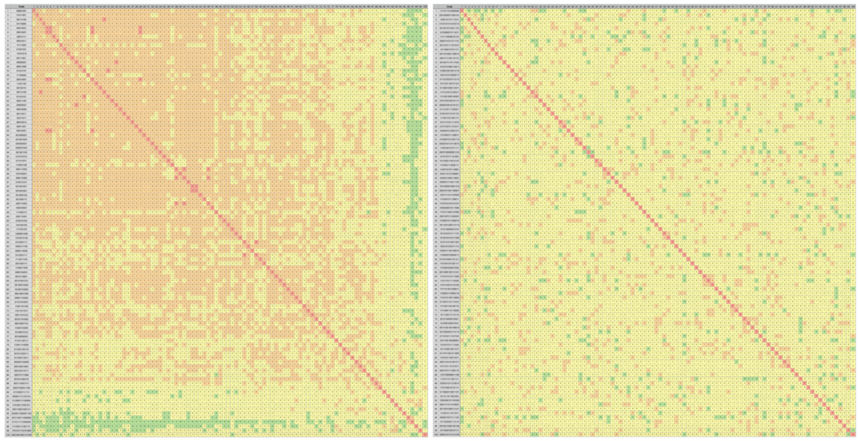

Figure 4: A pairwise computation of Levenshtein distances between user-generated (L) and randomly generated (R) codes.

Table III: User passcode's entropy per bit and nibble.

Group length	Bit Entropy	Nibble Entropy
8	0.9731	1.9296
8+	0.9865	1.9877

analysed how the last character, last two characters and last three characters influenced the selection of the next one in human-generated codes (see Table IV).

An interesting observation is that a significant proportion of the participants generated codes of the minimal length. Concretely, 13 out of 50 (or approx. 26%) generated codes of length 8, including the only repeated code ('00010001') within the training set.

Our working hypothesis is that users generating these shortest possible codes have an aversion to mental stress; not only do they produce shorter codes but also these codes tend to be less entropic when corrected for length. To test this, we computed the entropy per bit and nibble for all the 8 bits passcodes and all the 8+ ones. The results are shown in Table III.

Judging by the results in Table III, the hypothesis above seems to hold but not significantly enough as to justify the use of two different models (for 8 and 8+ codes) so we will use a single one, characterised by the transition matrix of Table IV.

We can use the probabilities shown in Table IV to implement a simple Markov model of the human-generated passcodes. The model will generate a passcode length following the distribution seen in the 50 codes of the training set, then randomly generate a first digit according to the $\{0.5735, 0.4264\}$ distribution of Table II, and then the rest until reaching the required length following the transition matrix of Table IV. It is interesting to note that despite the fact that the trigram model is supposed to be more accurate than the bigram one – and this should be preferred to the monogram model – in practical terms, we have seen that experimentally, all perform equally well with no statistically significant differences, so for simplicity, we used the monogram transition matrix in our study.

Table IV: User passcode transition matrix for monograms, bigrams and trigrams.

From → To	**0**	**1**
0	0.286	0.271
1	0.257	0.185
00	0.135	0.171
01	0.102	0.132
10	0.168	0.072
11	0.132	0.083
000	0.062	0.086
001	0.062	0.086
010	0.074	0.038
100	0.086	0.068
110	0.091	0.038
101	0.047	0.026
011	0.047	0.071
111	0.071	0.041

We used this model in our attack and measured its effectiveness against our test data of new, previously unseen 50 codes from a different set of participants. The results are in Figure 5. To compute this figure, we used the best minimum times achieved with the brute forcer, which is 150ms for a zero and 455ms for a one. In a nutshell, we can (on an average computed over 100 experiments) break around 7% of the passcodes after 1 minute, approximately 40% of the codes after 10 minutes, and around 75% of the passwords in the test dataset in 1 hour. On the other hand, for achieving a 95% we will need around 6 hours of code cracking with the brute forcer tool[7]. These figures correspond to the most realistic model of the Nokē, with the lockdown defence active. Otherwise, 2 hours would be enough to crack 93% of our test passwords.

[7] https://www.youtube.com/watch?v=IExycL7zif8

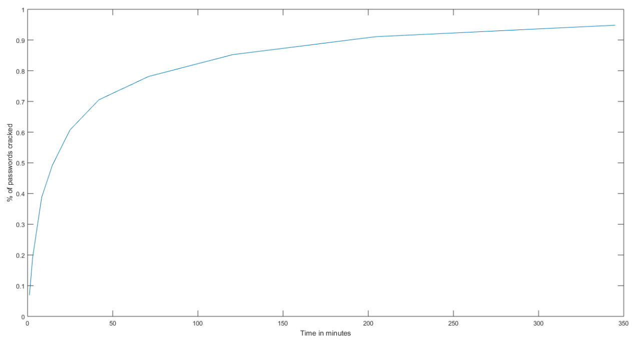

Figure 5: Percentage of cracked codes from the test dataset vs. Time in minutes.

VI Discussion

The interview stage, which involved data gathering from 100 participants, provided us with an interesting view of each user's rationale behind selecting their codes. We analysed the lengths and other patterns of these user-generated codes to come up with five categories.

VI-A Code Lengths and Patterns

From the code lengths that participants chose, it is interesting to point out that lengths 13 and 14 were infrequently used across the training and test data sets (see Figure 3). A length of 8, which is the shortest possible, was chosen by 29% of participants. Many of these cited memorisation related reasons to justify this length, indicating that shorter codes are easier to remember ("I'm worried I might forget my code" or "I picked a pattern that is easy to remember").

This data reinforced the importance of memorability as a key feature in user-defined heuristics; drawing parallels between manual overrides such as the quick-click code and other security metrics such as passwords, PINs or other numerical combinations. To further confirm this, the majority of the participants selected a code of length between 8 and 12, with 87% of the dataset in this group. We consider this to be a clear indication that a shorter and easier to remember code is better suited to the average user, and as a result, will be more frequently used than codes of maximum length; which, although better regarding security, are not always user-friendly.

Another trend we observed from our dataset was the participants' influences behind creating their quick-click codes. There are many instances in which these were related to songs, rhythmic cues or other structural repetitions, in other words, *earworms*. Participants described up to 46% (46/100) of codes as being inspired in this way. In larger datasets, we predict that this value will scale proportionally.

In some cases, participants chose a direct repetition of a pattern, by simply iterating it as many times as necessary to meet the desired length, e.g. ● ● ● - ● ● ● - or - ● - ● ● - ● - ● ● or ● ● ● - ● ● ● - ● ● ● or ● ● ● ● - - - ● ● ● ●.

We conjecture that in a larger sample, there will be a non-negligible number of (likely short) identical codes picked by different users (*birthday paradox*), probably of at least 5% of the codes. This is supported by our dataset, in which there were 5 repeated codes (all of length 8 or 9) leading to 10 duplicated entries.

General security management advice, and in particular the best practice and recommendations commonly employed for creating secure passwords, are tough to apply in the context of quick-click codes. Most of the readily available password generation guidelines (such as [23]) refer to having a mixture of characters and enforcing a minimum length; whilst the length of the quick-click code can be constrained, introducing variation to the sequence of presses is difficult given the limitation of having only long and short presses available. As a result, there is a relatively small number of unique combinations for the quick-click code ($130,816 = \sum_{n=8}^{n=16} 2^n$, or the equivalent of approximately 17 bits), which is very low when compared to other security metrics.

VI-B Five Categories of Patterns

After analysing the 100 codes – in particular, looking at their structural characteristics and the rationales given by the participants in their selection process – we classify the codes into five distinct categories, as can be seen in Table V.

The "Earworm" category was shown to be by far the largest in our dataset; with 46% of participants selecting a code of this nature – many of whom cited the catchiness of a given rhythmic sequence as a driving factor behind their choice ("The code correlated with a tune in my head I could remember"). Other common themes included Morse-like codes (we label them as "Morse" in Table V), the encoding of numerical data, such as birth dates and number sequences ("Numerical") and encoding strings of personal information relating to the participant ("Name Substitution").

Alternatively, 25% of our code sample was cited as being randomly chosen by participants ("Pseudo-random"). In some cases, this occurred where the participant, with the intention to make it "harder to guess" devised a seemingly complex and structureless code. Our results also show that 11 out of 29 of the minimal code length of 8 (38%) belonged to this category.

VI-C Other Issues

A major issue we found in the Nokē is that the quick-click code is a mandatory feature which cannot be disabled by the user under any circumstances. The problem here resides in the fact that the system compels the user into using the code; which may result in poorly thought-out, quick and dirty heuristics being used in place of a proper, secure code.

Considering the lack of variations in the quick-click code alphabet (either a long press or a short press) this is conducive to low entropy passwords, increasing the likelihood

Table V: Five categories of quick-click codes, sorted by their frequencies (n=100).

Category	Description	Freq.
Earworm	Codes related to music, rhythmic cues or other structural repetitions.	46
Pseudo-random	Codes which are seemingly unstructured, or cited by participants as being randomised.	25
Numerical	Codes which feature numerical encodings, such as a favourite number or date of birth.	19
Morse	Codes based on the encoding of words or phrases in Morse code.	8
Name Substitution	Codes related to the participant's name, e.g. by encoding vowels and consonants as presses.	2

that the correct codes are guessed quickly through semi-randomised attempts. For an attack of this nature, it is possible that brute forcing devices manipulating the shackle could be a useful tool in the attacker's arsenal to target individual Nokē devices with a series of automated attempts. This would be particularly effective if there were little in the way of preventative measures in the device, as successive code entry attempts could then be chained together to recover the passcode at an increased rate.

Other IoT devices may employ different techniques, such as a factory reset or pairing up with a new smartphone, to deal with the possibility of the associated smartphone becomes unavailable. Nevertheless, reduced entropy input is common with smart locks, e.g. MasterLock uses directional combination padlock and ResLock employs a dedicated button for a Morse-code. Furthermore, many smart locks designed for home security, such as the Schlage Connect, feature numerical override systems built on user input, such as a four digit passcode. These systems will also suffer from weak entropy, and they are prone to our attack.

According to our model, 40% of the codes can be cracked under 10 minutes. The success of a real world attack based on this (e.g. for stealing a bike) depends on several factors. Against more permanent targets, such as a storage container or a shed our model indicates a 93% success rate after running the brute forcer for 2 hours.

VI-D Potential Countermeasures

Our research has shown that smart locks with a manual override, such as the Nokē, open themselves to a potential vulnerability that can significantly reduce their overall security. We propose several countermeasures to improve their security (at the cost of reduced usability):

- Provide an option for the user to disable the quick-click override mechanism. This would remove the risk of a brute force attack, but the user might not be able to open the lock if they lost their phone.
- Employ an additional movement (e.g. a different shackle pressure) or add a button to indicate the beginning of the quick-click code entry explicitly.
- Do the same (or even better, use a different marker) to signify the end of the quick-click code explicitly.
- Remove the light/sound feedback when the user enters the right code, as this eases automatic success detection.
- Implement immediate penalisation delays after every failed attempt, or even a staggered increase of the delay after repeated delayed attempts, as in the French passport [24]. The current implementation allows users to enter 64 wrong quick-click codes in succession before triggering a lockout period of 1 minute. A single attempt with 64 quick-click codes can potentially be used to test up to 1,632 keys (not all necessarily different), which of course simplifies the attacker's task quite significantly.
  In general, triggering this lockout penalisation mechanism after $m > 8$ quick-click attempts allows the attacker to try up to $\sum_{i=8}^{m}(m-i)+1 = m\cdot(m-8)+\frac{(m+8)(m-8)}{2}+m$ keys, so it is very important for security reasons to keep these m as small as possible without compromising usability too much. For example, if the value of m is halved from the current value of $m = 64$, the number of these spurious key trials will be reduced by a factor of four, from 1,632 to 352. A further reduction to 16 would increase security (up to a maximum of 96 spurious keys), but will likely compromise usability too much. We recommend reducing m to a value between 24 and 32.

These series of simple improvements would have a massive impact in reducing the effectiveness of our brute forcer tool against the Nokē and similar smart-locks.

VII Conclusion

In this study, we investigated several aspects of user behaviour when choosing a personal security metric such as an override code for IoT smart locks. We have also compared the statistical aspects of user-based heuristics for selecting passcodes against those that are automatically generated by the system and analysed patterns related to the selection of these codes.

Based on the statistical analysis of the collected human-chosen codes from 100 participants, we conclude that there is a serious vulnerability in current implementations of smart lock manual overrides, which would enable a brute force attack to guess the passcode with very high probability in only minutes.

The reasons behind this vulnerability include:

- Small range of unique code possibilities. For codes of length 8, there are 256 unique combinations to choose from, and just 65,536 for the maximum length of 16

– both of which are relatively small compared to other security metrics (e.g. passwords).

- User behaviour reflects a stark inclination to choose shorter length codes, with 29% opting for the minimal length of 8.
- Users are heavily biased towards using more zeros than ones in their codes, which further speeds up the attack as zeros are 3x faster to click.
- The use of visible patterns and repetitions is a common occurrence in our data which, severely affects the entropy of the used codes.
- There is no option to disable the override feature, and there are only limited measures in place to prevent or detect attackers manipulating the device.

To demonstrate the feasibility of a brute force attack on the manual override mechanism, we implemented a mechanical brute forcer using information from our models. The brute forcer takes 2.52 seconds on average to test a single passcode, which is a significant advantage over a human performing the same task manually (which in our investigation has been empirically estimated at between 2-4 times slower than the brute forcer).

We proposed several countermeasures to mitigate this vulnerability, but they have a cost regarding usability. This highlights the need to explore further issues relating to security and usability trade-offs in emerging technologies such as IoT.

VIII FUTURE WORK

A promising area of our future work is the development of a standard for vendors' implementation of manual overrides in smart locks. There is a lack of consistency between the levels of security offered by different vendors; for example, Masterlock has a feature where the legitimate owner is sent a warning email when the manual override code is incorrectly entered upon repeated attempts. Although the Masterlock primary code is considerably shorter at a fixed length of 7, this feature reinforces the user's awareness of the device's status and serves as a useful tool in preventing theft. The Nokē, and other future smart locks could benefit greatly from a similar system alerting users to any suspicious activity.

Finally, to build upon the findings from our research, we would like to expand the size of our dataset by collecting more samples from a wider target population. Whilst the current results successfully demonstrate patterns in the quick-click code selection, with a larger participant sample we would be able to observe more trends in the data and increase the likelihood that they could be applied to the wider user group of smart locks, and IoT devices in general.

ACKNOWLEDGEMENTS

We thank the group of participants who took part in our study and shared their experience of using the Nokē smart lock, and the anonymous reviewers for their useful feedback.

REFERENCES

[1] Brian Krebs. Krebs On Security Hit With Record DDoS. *https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/*, 2016.

[2] Dan Goodin. Brace yourselvessource code powering potent IoT DDoSes just went public. *http://arstechnica.com/security/2016/10/brace-yourselves-source-code-powering-potent-iot-ddoses-just-went-public/*, 2016.

[3] Eyal Ronen and Adi Shamir. Extended functionality attacks on IoT devices: The case of smart lights. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 3–12. IEEE, 2016.

[4] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. Security analysis of emerging smart home applications. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 636–654. IEEE, 2016.

[5] Charlie Miller and Chris Valasek. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA*, 2015, 2015.

[6] Dae Gyu Seo, Han Shin Ko, and Yong Deok Noh. Design and Implementation of Digital Door Lock by IoT. *KIISE Transactions on Computing Practices*, 21(3):215–222, 2015.

[7] Kishore Angrishi. Turning Internet of Things (IoT) into Internet of Vulnerabilities (IoV): IoT Botnets. *arXiv preprint arXiv:1702.03681*, 2017.

[8] Eyal Ronen, Colin OFlynn, Adi Shamir, and Achi-Or Weingarten. IoT Goes Nuclear: Creating a ZigBee Chain Reaction. *https://eprint.iacr.org/2016/1047.pdf*, 2016.

[9] Gaja Kochaniewicz. Smart lock for bike sharing in corporate environments. Master's thesis, 2015-06-10.

[10] T. Valerio and S. Payne. Padlock, December 10 2009. US Patent App. 12/515,296.

[11] Matt Blaze. Safecracking for the computer scientist. *U. Penn CIS Department Technical Report*, 2004.

[12] L.B. Ranchod. Keyless padlock, system and method of use, August 18 2015. US Patent 9,109,379.

[13] Masterlock. Masterlock Bluetooth Padlock, April 2017.

[14] Fuzdesigns. Noke Quick Click Code, April 2017.

[15] Theo Tryfonas and Ioannis Askoxylakis. *Human Aspects of Information Security, Privacy, and Trust: Third International Conference, HAS 2015, Held as Part of HCI International 2015, Los Angeles, CA, USA, August 2-7, 2015. Proceedings*, volume 9190. Springer, 2015.

[16] Alain Forget and Robert Biddle. Memorability of Persuasive Passwords. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '08, pages 3759–3764, New York, NY, USA, 2008. ACM.

[17] Anne Adams, Martina Angela Sasse, and Peter Lunt. Making passwords secure and usable. In *People and Computers XII*, pages 1–19. Springer, 1997.

[18] Steffen Werner and Connor Hoover. Cognitive approaches to password memorability–the possible role of story-based passwords. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 56, pages 1243–1247. SAGE Publications, 2012.

[19] Jeff Jianxin Yan, Alan F Blackwell, Ross J Anderson, and Alasdair Grant. Password Memorability and Security: Empirical Results. *IEEE Security & Privacy*, 2(5):25–31, 2004.

[20] Sascha Fahl, Marian Harbach, Yasemin Acar, and Matthew Smith. On the ecological validity of a password study. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*. ACM, 2013.

[21] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, 1999.

[22] Michael Gilleland. Levenshtein distance, in three flavors. *Merriam Park Software: http://www.merriampark.com/ld.htm*, 2009.

[23] Matteo Dell'Amico, Pietro Michiardi, and Yves Roudier. Password Strength: An Empirical Analysis. In *INFOCOM*, volume 10, pages 983–991, 2010.

[24] Tom Chothia and Vitaliy Smirnov. A traceability attack against e-passports. In *International Conference on Financial Cryptography and Data Security*, pages 20–34. Springer, 2010.