CS4203 P02 - Practical Applications

150015673

18 November 2019

Part 1: [813 words]
Part 2: [1970 words]

Contents

1 Penetration Testing

1.1 Introduction

The aim of this section was to do a passive evaluation of two website's security using various tools (details in Appendix A). The first website was `bbc.co.uk` and for the second I chose `borger.dk` (literally 'citizen.dk'), which is the Danish government's online portal for Danish citizens to manage most public services, e.g. healthcare, study loans, pensions, and many more (an English version of the website is available here).

1.2 bbc.co.uk

Registration Details				
Registrar	Registered on	Expires		
British Broadcasting Corporation	before August 1996	13 Dec 2025		
Servers	**Servers**			
Name servers	ns3.bbc.co.uk (IPv4)	ns3.bbc.net.uk (IPv6)	ns4.bbc.co.uk (IPv4)	ns4.bbc.net.uk (IPv6)
Mail servers	cluster1.eu.messagelabs.com	cluster1a.eu.messagelabs.com		
Ports				
	Status	Usage	notes	
25/tcp	closed	smtp		
53/tcp	closed	domain		
80/tcp	open	http-proxy	redirects to https://bbc.co.uk	
111/tcp	closed	rpcbind		
135/tcp	closed	msrpc		
139/tcp	closed	netbios-ssn		
443/tcp	open	ssl/https-proxy		
445/tcp	closed	microsoft-ds		
2049/tcp	closed	nfs		
31337/tcp	closed	Elite	likely closed to deny a common attack port	
Operating System guesses				
Name	Probability	Details		
Linksys embedded	92%	(BEFW11S4 WAP: 92% BEFSR41 router 89%)		
Asus embedded	88%	(RT-53N WAP 88%)		
Cisco embedded	86%	(ACE load balancer 86%)		

Linux 2.6.X	85%	(Linux 2.6.11 - 2.6.18 85%)		
Vulnerabilities				
Source	Comments			
Linksys	most are 14-15 years old and so hopefully patched [8]			
MessageLabs (Symantec)	there are a couple from November 2019 detailing antivirus bypasses [9]			
Cross-Domain JS src included	all are from the `bbci` domain; they are home-brewed			
XSS Browser Protection not enabled	XSS HTTP response is deprecated [5]			
`X-Content-Type-Options` not set to `nosniff`	unsure how common MIME-sniffing attacks are, but it should be set [4]			

Table 1: Information gathered about bbc.co.uk

1.3 Discussion

Fortunately, it seems most of the services used to host bbc.co.uk are secure. The Linksys access points or routers had some vulnerabilities according to the NIST NVD [8], but these are 14-15 years old and therefore hopefully fixed today. The `messagelabs` domain used for the mail servers is a European email company which was acquired by Symantec [6]. They are one of many companies which offer corporate emails with screening/increased security checks of the emails sent. However, Symantec cloud (which they are part of [6]) had a couple of vulnerabilities in November 2019 detailing antivirus bypasses [9], which is problematic if that is the main point of defence against potentially malicious emails. When running OWASP ZAP, it flagged the Content Security Policy (CSP) as missing a couple of fields. By looking at the Mozilla Developer Network (MDN), I found out that the first field (XSS Browser Protection) is a lingering, deprecated artefact from Internet Explorer, which all other browsers either do not support or have phased out [4]. The second field (`X-Content-Type-Options`) protects against so-called “MIME type sniffing” where a user tries to upload a malicious file by disguising its type as something else [17]. I was unable to find information on whether these attacks are still common, but the MDN notes state that most security testers will expect the field to be set [5], so it likely should be. The email server vulnerabilities might be a problem if they have not yet been patched, but apart from that, there does not seem to be any major security risks.

1.4 borger.dk

Registration Details			
Registrar	Registered on	Expires	
Statens IT (the state's IT)	7 Jan 2003	31 Jan 2020	
Servers			
Name servers	ns2.sitnet.dk	ns3.sitnet.dk	
Mail servers	mx5.sitnet.dk	mx6.sitnet.dk	
Ports			
	Status	Usage	notes
25/tcp	closed	smtp	
53/tcp	closed	domain	
80/tcp	open	http	redirects to https://borger.dk
111/tcp	closed	rpcbind	
135/tcp	closed	msrpc	
139/tcp	closed	netbios-ssn	
443/tcp	open	ssl/http	Uses "Citrix NetScaler"; redirects to https://borger.dk
445/tcp	closed	microsoft-ds	
2049/tcp	closed	nfs	
31337/tcp	closed	Elite	likely closed to deny a common attack port
Operating System guesses			
Name	Probability		
Liksys BEFW11S4 WAP	93%		
Cisco Adaptive Security Appliance 5510 or 5540 fire-wall	89%		
Cisco C2960 Switch (IOS 12.2)	89%		

Linksys BEFSR41 router	89%		
Vulnerabilities			
Source	Comments		
Content Security Policy	the `script-src` whitelist can be bypassed [3]; `unsafe-inline` allows any inline scripts and event-handlers to execute, especially as it does not include a nonce [3]; furthermore, the lack of an `object-src` rule allows script execution by injecting plugin resources [15]; no `frame-ancestor` rule definition and no indication of mitigation in the source code, meaning the site could potentially be used for “clickjacking” [10];		

Table 2: Information gathered about borger.dk

1.5 Discussion

The borger.dk website seems to use the same Linksys system as the BBC discussed above. Apart from that, the main, much more serious problem seems to be the website’s CSP. By using an online CSP-evaluator made by Google [3] and by reading an extensive study of CSP policies by Weichselbaum et al. [15], I found several problems with the website’s CSP (the full CSP is in Appendix B). The lack of a `frame-ancestor` rule with no prevention in the source-code either, the website is vulnerable to clickjacking [3, 10, 13]. As the site has a login button, clickjacking could be highly problematic by having an attacker link to a malicious cites on top of a seemingly legitimate button [13]. There are also problems with the scripting rules in the CSP: `unsafe-inline` is specified and there is no `object-src` rule. The first would be okay if the server associated a nonce to each script to certify them [16]. However, no such rule is present, thereby potentially allowing any script to execute [3]. The second problem, the lack of an `object-src` rule, means that a library or extension could potentially be injected and execute malicious JavaScript code [3, 7, 15]. Finally, I was unable to find out how easy or difficult it is to do, but according to [3], the whitelist of trusted script source code could be bypassed, defeating the rule. In combination with the previous things discussed, this would allow for arbitrary code loading and execution. In conclusion, borger.dk seems to have some serious security problems for a site whose functionality is at least as critical as online banking.

2 Rhythmic Keylogger for Authentication

2.1 Introduction

Passwords (PWDs) are not as secure as people believe, especially as people are likely to use common words or phrases, thereby making them vulnerable to dictionary attacks. One theory is that it is possible to strengthen passwords by also checking whether it matches the user's typing rhythm, i.e. the time they usually take between keystrokes, how many keystrokes they make in a "burst", etc. The main potential benefit of this theory is that adding it to an existing authentication solution would not require major changes to the software or hardware: the keyboard and timings of a user typing happen anyway, the system would just need to monitor them.

2.2 Problem Statement and Hypotheses

By implementing a simple keylogger, this section aims to do an initial, alpha (i.e. self-tested) study of the theory mentioned above. Specifically, the following four hypotheses (three from the specification and one personal) will be examined:

1. *A simple rhythm can be determined more easily than a more patterned or dictionary-based one.*
2. *Some patterns are more easily detected or broken than others.*
3. *There is an effect from the ID entry rhythm, specifically that its shortness may be a factor and affects overall false-positive acceptance of the ID-PWD pair.*
4. Through continued use of the same ID-PWD pairs, muscle memory eventually changes the observed pattern significantly.

2.3 Relevant Background

The idea of using typing rhythms and patterns to harden username and password authentication is far from new. Over two decades ago, de Ru and Eloff demonstrated a system which used typing patterns in addition to username and password to authenticate users [12]. However, the paper also acknowledged a problem: People's typing patterns change, especially as repeated sequences (e.g. a password) become more ingrained in muscle memory. As such, some form of continuous learning and flexibility of the system is required, which may be computationally expensive [12].

Later studies [2, 14] have confirmed the notion that typing patterns reinforce password-based authentication. The main difficulty seems to be the question of how to obtain the model to derive the patterns from [2]. Teh et al. use a system where the user has to enter their username and password 10 times, and then the phrase "the brown fox" [14] and also implement a retrain functionality to account for changing typing habits.

2.4 Methodology and Experiment

The setup for the alpha study was as follows:

- the machine was running Linux 5.3.7
- input to the OS was handled using the `Xorg X server v1.20.5` and `xf86-input-libinput v0.29.0`
- the user was proficient in touch typing

- the user's keyboard layout was the Norwegian Dvorak layout (see Appendix C)

The test cases input are listed in the table below:

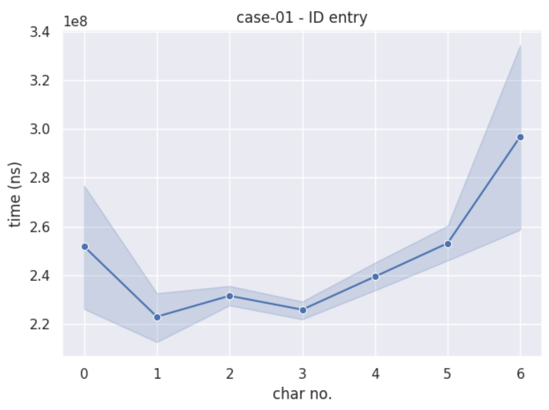
Test Case	ID	Password	Notes
1	AAAAAA	123456	Baseline pattern
2	AAAAAA	a1a1a1	Simple pattern
3	AaAaAaAa	a1a1a1	Timing effect of 1 element change
4	AAAAAA	c0mput3r_sc13nc3	Timing effect of complex pwd
5	Å,.PYF	a2b3c4d5	Fast speed of ID effect
6	Å,.PYF	mbxkjq	Fast speed of ID effect
7	Azbycxdw	wdxcybza	Complex memory task
8	Azbycxdw	nthdiu12	Faster password effect
9	æqjkxbm	å,eukxdhcr	Moving rightwards (right hand)
10	mbxkjqæ	lrthbxiup.	Moving leftwards (left hand)
11	root	password	Common combination
12	root	c0mput3r_sc13nc3	Short ID, long password
13	user	password	Common combination
14	user	c0mput3r_sc13nc3	Short ID, long password
15	guest	password	Common combination
16	guest	c0mput3r_sc13nc3	Short ID, long password

Table 3: Sample user IDs and passwords

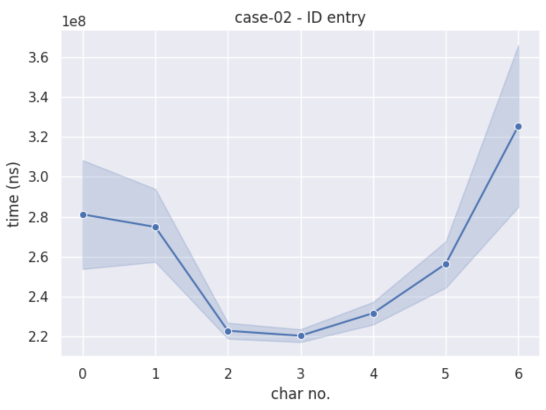
In cases 5 and 6, the ID entry was performed with `[CapsLock]` enabled. In all other cases, capital letters were entered by also holding down the `[Shift]` key.

Each case was entered and its associated data collected 10 times. The data collected was then plotted with standard error mean (SEM) to produce analysable graphs. The code used can be found in Appendix E. The first data-point in each plot starts high up on the curve due to waiting for the keylogger to start, or to confirm that the previous entry had been recorded (a slight pause after hitting `[Enter]`).

2.5 Results



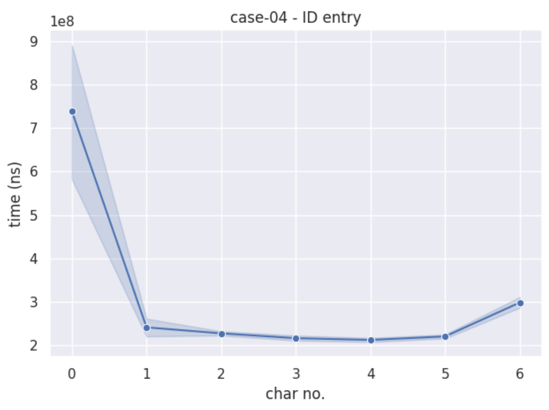
(a) Key timings for ID 1



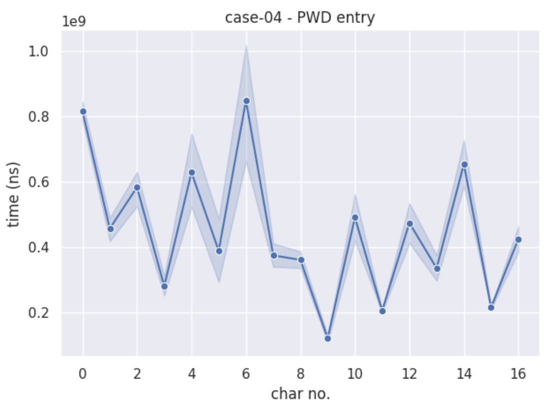
(b) Key timings for ID 2

Figure 1: Timing comparison for entering ID 1 and 2

Comparing the timings for entering the ID in cases 1 and 2, there is a similarity at the end where they both increase in time (Figure 1). This is due to the user pausing to count 'A's before entering the final one. Interestingly, when looking a case 4 which has the same ID as cases 1 and 2, the pattern has completely changed (Figure 2), possibly due to muscle memory or greater confidence in typing the right number of 'A's. It does, however, still exhibit a slight increase in time at the end, indicating that the user is still pausing to confirm the final A is not one too many (there are 7 entries compared to 6 As due to `[Shift]` counting as a keystroke).



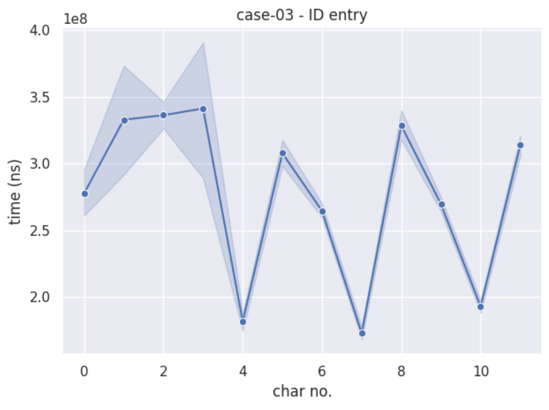
(a) Key timings for ID 4



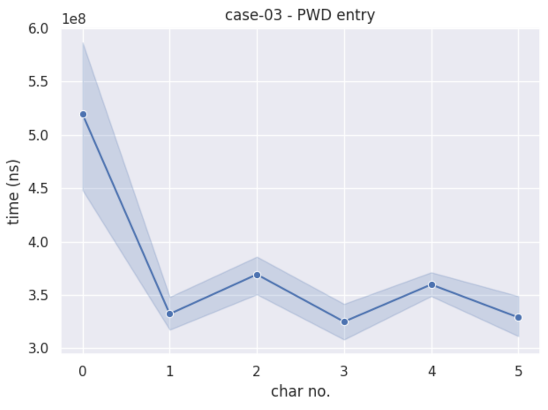
(b) Key timings for PWD 4

Figure 2: Results from case 4's ID and PWD combination

When examining the ID and PWD combination from case 3, there is a "saw-tooth" pattern (Figure 3), which is likely caused by the alternating characters in the ID and the PWD. There is one extra keystroke in the saw-tooth pattern of the ID due to pressing `[Shift]` to enter a capital `A`. When entering the case 3 credentials, I found myself following a “`tA-ta-tA-ta-tA-ta-tA-ta`” rhythm, which corresponds to the pattern.



(a) Key timings for ID 3



(b) Key timings for PWD 3

Figure 3: Results from case 3’s ID and PWD combination

Case 2, which has the same password as case 3, has a flat line initially, with the saw-tooth pattern seemingly beginning in the final 3 keystrokes (Figure 4). I do not know what I managed to consistently do different when entering case 2’s password. One explanation could be that the different IDs affected my typing pattern in some way.

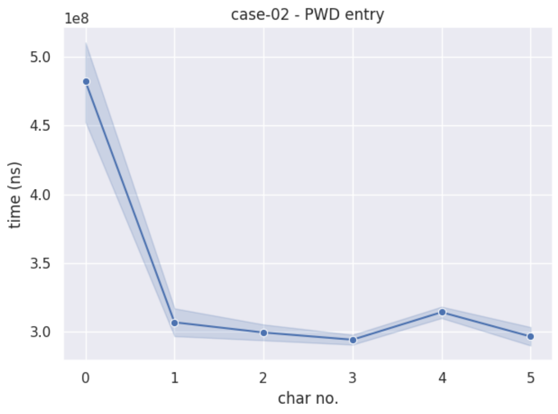
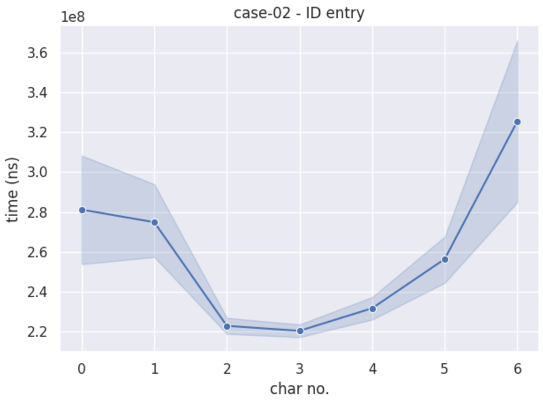
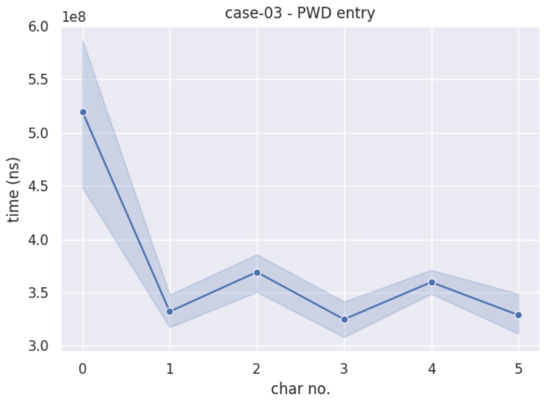


Figure 4: Key timings for PWD 2

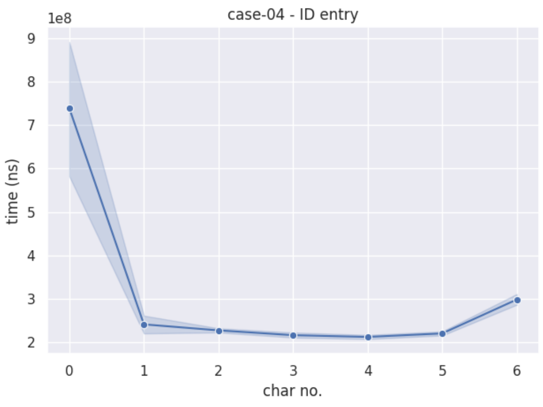
Comparing simple patterns from the IDs in cases 1, 2, and 4 (Figures 1 and 2), with patterned rhythms from the ID in case 3, passwords in cases 2 and 3 (alternating character pattern; Figures 3 and 4), and passwords in cases 9 and 10 (going across the keyboard in a “staircase” pattern, see Appendix C: Figure 12), the data suggests little difference in the SEM between simple patterns and patterned rhythms (Figure 5). The ID entry in cases 1, 2, and 4 (Figures 1 and 2) highlight that a change occurs at some point, probably due to muscle-memory improving (which would support hypothesis 4). Additionally, cases 5-8 (omitted here for brevity; see Appendix D) indicate that the distance between keys affects the overall time, especially if the sequence is one that can be "rolled over" in one motion. Interestingly, cases 7 and 8 (Appendix D: Figures 15 and 16 respectively) show that a more complex ID or password does not necessarily lead to a more complicated pattern. This could be because of the left-hand/right-hand alternating pattern enabled by the Dvorak layout. The cases discussed in this section seem to support hypothesis 1: the more complex patterns would likely be more difficult to try to imitate, especially those which have a more complex timing pattern with consistently low SEM (e.g. PWD 4; Figure 2).



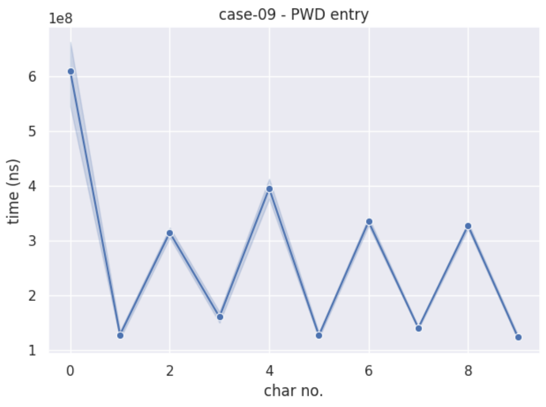
(a) Simple pattern (ID 2)



(b) Patterned rhythm (PWD 3)



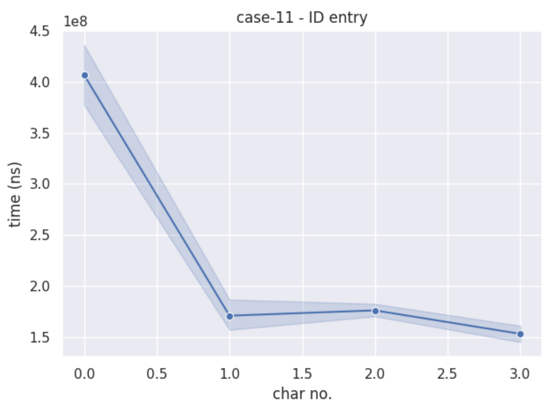
(c) Simple pattern (ID 4)



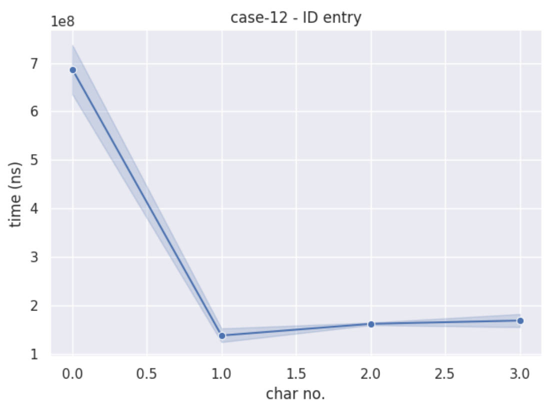
(d) Patterned rhythm (PWD 9)

Figure 5: Comparison of simple patterns and patterned rhythms.

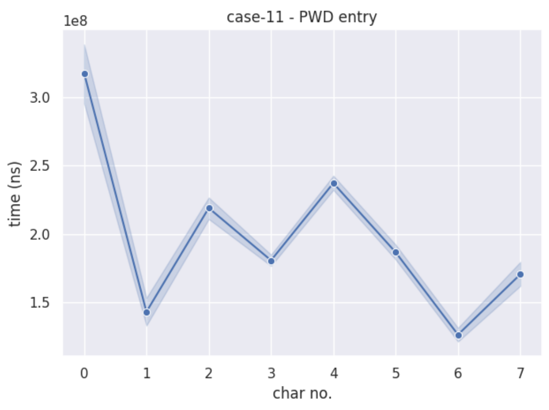
Further looking at more dictionary-based password rhythms, like cases 4 (Figure 2) and 11 (Figure 6), the SEM is still low, indicating that the rhythm is consistent from sample to sample. Contrary to the simpler saw-tooth patterns, the example patterns from the passwords of cases 4 (Figure 2) and 11 (Figure 6) are more complex and therefore would likely be more difficult to mimic or break.



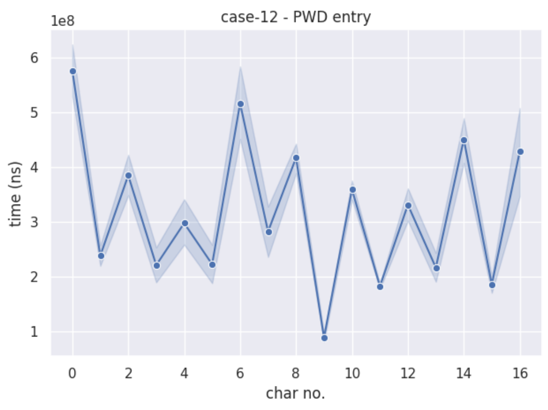
(a) Key timings for ID 11



(b) Key timings for ID 12



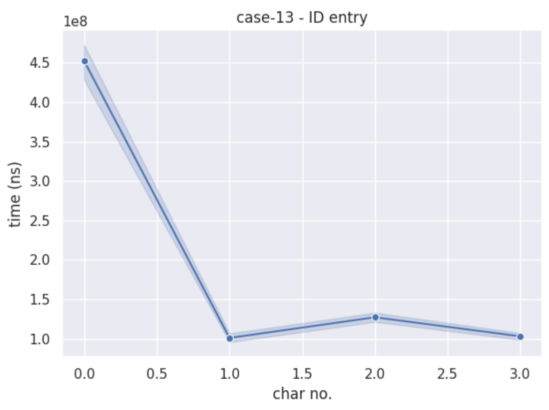
(c) Key timings for PWD 11



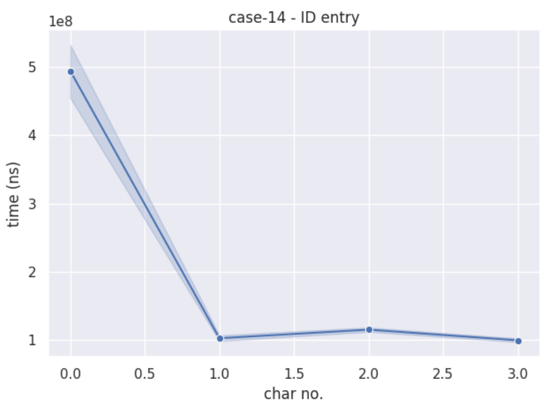
(d) Key timings for PWD 12

Figure 6: Results for cases 11 and 12

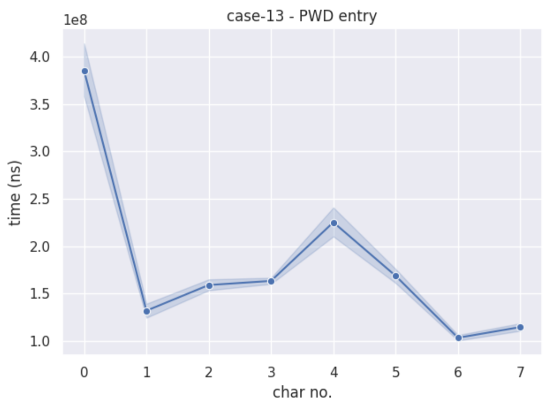
Finally, examining cases 4 and 11-16 (Figures 2, 6, 7, and 8) does not seem to indicate a correlation between short ID and the strength of the ID-PWD pair, contrary to hypothesis 3. With a short ID, the timing patterns have a very low SEM, likely caused by the simplicity of the ID. This could mean that it is difficult to mimic the ID-PWD combination precisely enough for a false-positive to register.



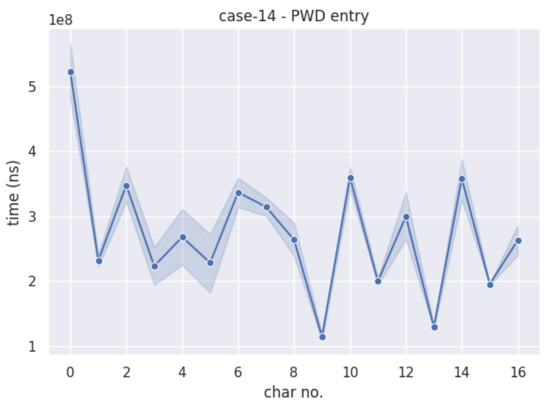
(a) Key timings for ID 13



(b) Key timings for ID 14



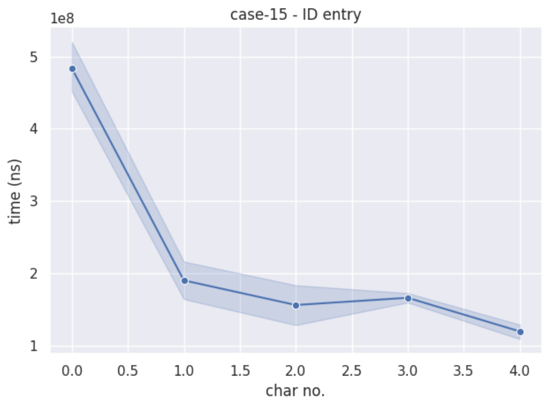
(c) Key timings for PWD 13



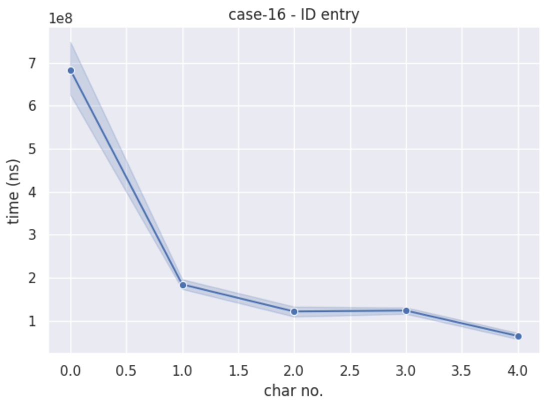
(d) Key timings for PWD 14

Figure 7: Results for cases 13 and 14

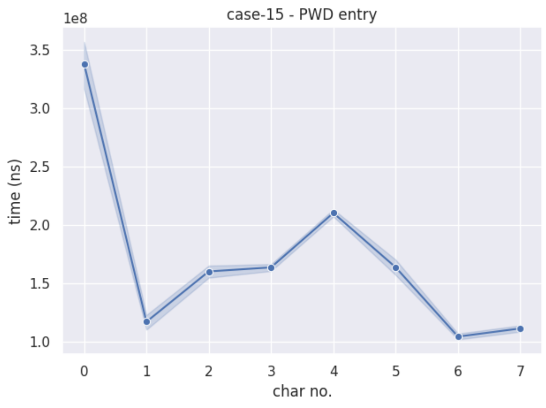
The same cases seem to be indicative of the ID text affecting the password curve and, even more interestingly, show a clear point after which muscle memory has changed the typing pattern, supporting the 4th hypothesis.



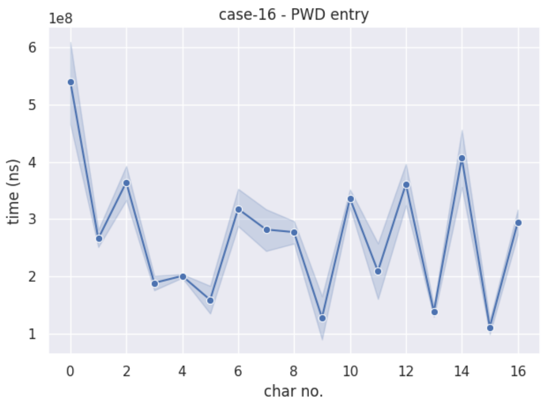
(a) Key timings for ID 15



(b) Key timings for ID 16



(c) Key timings for PWD 15



(d) Key timings for PWD 16

Figure 8: Results for cases 15 and 16

2.6 Conclusion

A keylogger that registered the timing/typing pattern of a user entering their ID and PWD was successfully implemented. Through data collection and analysis of 16 sample ID-PWD combinations, the four hypotheses presented in sub-section 2.2 have been examined. Based on the collected data, hypotheses 1 and 2 seem to hold true, i.e. the simple rhythms are easier to determine than their more complex/varied counterparts and some patterns, e.g. the ones which can be typed by moving 'in one direction' on the keyboard, might be easier to detect or break. However, contrary to hypothesis 3, the short length of the ID entry rhythm does not suggest to significantly weaken the ID-PWD pair, due to the much tighter tolerance/SEM that a short ID has. In order to fully confirm or deny these hypotheses, the main limitation of this study would need to be addressed: all statements are based on the timings collected and not on data from a system which was trained on the collected timings. This would give stronger data in the form of false-/true-positives and -negatives. However, such an authentication system would require an AI or Machine Learning approach, and was therefore deemed out of scope for this alpha study. Comparing early and later examples of the same ID or password, hypothesis 4 seems to be true. Specifically, the change appears to occur around the 30th entry of the character sequence. All in all, in agreement with the existing literature, typing patterns seem to be a good and reasonably easy way to harden the existing, highly popular ID-PWD authentication method.

2.7 Future work

As this is an alpha study, there are many opportunities for further research. The results from IDs 11-16 show that whilst the curves have a similar shape, each covers a different timing range. More research could be done focusing on the short IDs, for instance on whether people eventually develop similar enough short burst typing patterns to cause false-positive timing patterns for different ID texts.

Another interesting question is whether, and by how much, the ID text affects the password timing. By the changes in curve in cases 11, 13, and 15 and cases 12, 14, and 16 (which have the same password), it seems the ID text might change the password timing pattern (possibly due to different starting positions of the fingers after typing the ID) but further research is needed to conclude whether it is the muscle-memory affecting the curve or the ID text.

Research could also be conducted into capturing other aspects of someone's typing rhythm, e.g. how often they delete a character when typing. A recent study by Alsultan et al. [1], examined the use of "non-conventional" typing data, e.g. editing patterns, for profiling and *continuously* authenticating users. By analysing free-text data (e.g. emails and documents) instead of fixed-text data (e.g. usernames and passwords), they were able to obtain low false acceptance and false rejection rates thereby suggesting that there may be benefits to collecting data beyond the timing of (individual) keystrokes. It would be interesting to see how, or if, the collection of non-conventional typing data improves the accuracy of fixed-text authentication like the ID-PWD combination examined in this practical.

Finally, the increasing popularity and adoption of password managers [11] raises the question of whether typing-pattern-based hardening of authentication is emerging too late, or if the two technologies can be combined in some way. It may be that as 25-30 character passwords increase in popularity, through auto-generation by password managers, the necessity for this hardening decreases. However, as OS authentication does not support password manager auto-completion and most password managers have a notion of a 'master password' to access the stored passwords, it is possible that the hardening could be applied to these cases. In any case, password managers are gaining popularity and their place in hardening authentication could be interesting to see.

References

[1] Alsultan, A. et al. “Non-conventional keystroke dynamics for user authentication”. In: *Pattern Recognition Letters* 89 (Apr. 2017), pp. 53–59. ISSN: 01678655. DOI: 10.1016/j.patrec.2017.02.010. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167865517300429 (visited on 11/12/2019).

[2] Chen, W. and Chang, W. “Applying hidden markov models to keystroke pattern analysis for password verification”. In: *Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration, 2004. IRI 2004.* 2004 IEEE International Conference on Information Reuse and Integration, 2004. IRI 2004. Las Vegas, NV, USA: IEEE, 2004, pp. 467–474. ISBN: 978-0-7803-8819-2. DOI: 10.1109/IRI.2004.1431505. URL: http://ieeexplore.ieee.org/document/1431505/ (visited on 11/12/2019).

[3] GOOGLE. *CSP Evaluator*. URL: https://csp-evaluator.withgoogle.com/.

[4] *MDN - X-Content-Type-Options*. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options.

[5] *MDN - X-XSS-Protection*. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection.

[6] *MessageLabs*. Crunchbase. URL: https://www.crunchbase.com/organization/messagelabs.

[7] *Missing object-src in CSP Declaration*. Netsparker. URL: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-object-src-in-csp-declaration/.

[8] *NVD - Linksys BEFW*. NIST National Vulnerability Database. URL: https://nvd.nist.gov/vuln/search/results?form_type=Basic&results_type=overview&query=linksys+befw&search_type=all.

[9] *NVD - Symantec Cloud*. NIST National Vulnerability Database. URL: https://nvd.nist.gov/vuln/search/results?form_type=Basic&results_type=overview&query=symantec+cloud&search_type=all.

[10] *OWASP - Clickjacking*. Open Web Application Security Project. Dec. 21, 2017. URL: https://www.owasp.org/index.php/Clickjacking.

[11] *Password Managers: Under the Hood of Secrets Management*. Independent Security Evaluators. Feb. 19, 2019. URL: https://www.ise.io/casestudies/password-manager-hacking/.

[12] Ru, W. de and Eloff, J. “Enhanced password authentication through fuzzy logic”. In: *IEEE Expert* 12.6 (Nov. 1997), pp. 38–45. ISSN: 0885-9000. DOI: 10.1109/64.642960. URL: http://ieeexplore.ieee.org/document/642960/.

[13] Sejpal, A. *Why am I anxious about Clickjacking?* Dec. 2, 2014. URL: https://www.linkedin.com/pulse/20141202104842-120953718-why-am-i-anxious-about-clickjacking (visited on 11/15/2019).

[14] Teh, P. S. et al. “Keystroke dynamics in password authentication enhancement”. In: *Expert Systems with Applications* 37.12 (Dec. 2010), pp. 8618–8627. ISSN: 09574174. DOI: 10.1016/j.eswa.2010.06.097. URL: https://linkinghub.elsevier.com/retrieve/pii/S0957417410006019 (visited on 11/12/2019).

[15] Weichselbaum, L. et al. “CSP Is Dead, Long Live CSP! On the Insecurity of Whitelists and the Future of Content Security Policy”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS’16.* the 2016 ACM SIGSAC Conference. Vienna, Austria: ACM Press, 2016, pp. 1376–1387. ISBN: 978-1-4503-4139-4. DOI: 10.1145/2976749.2978363.

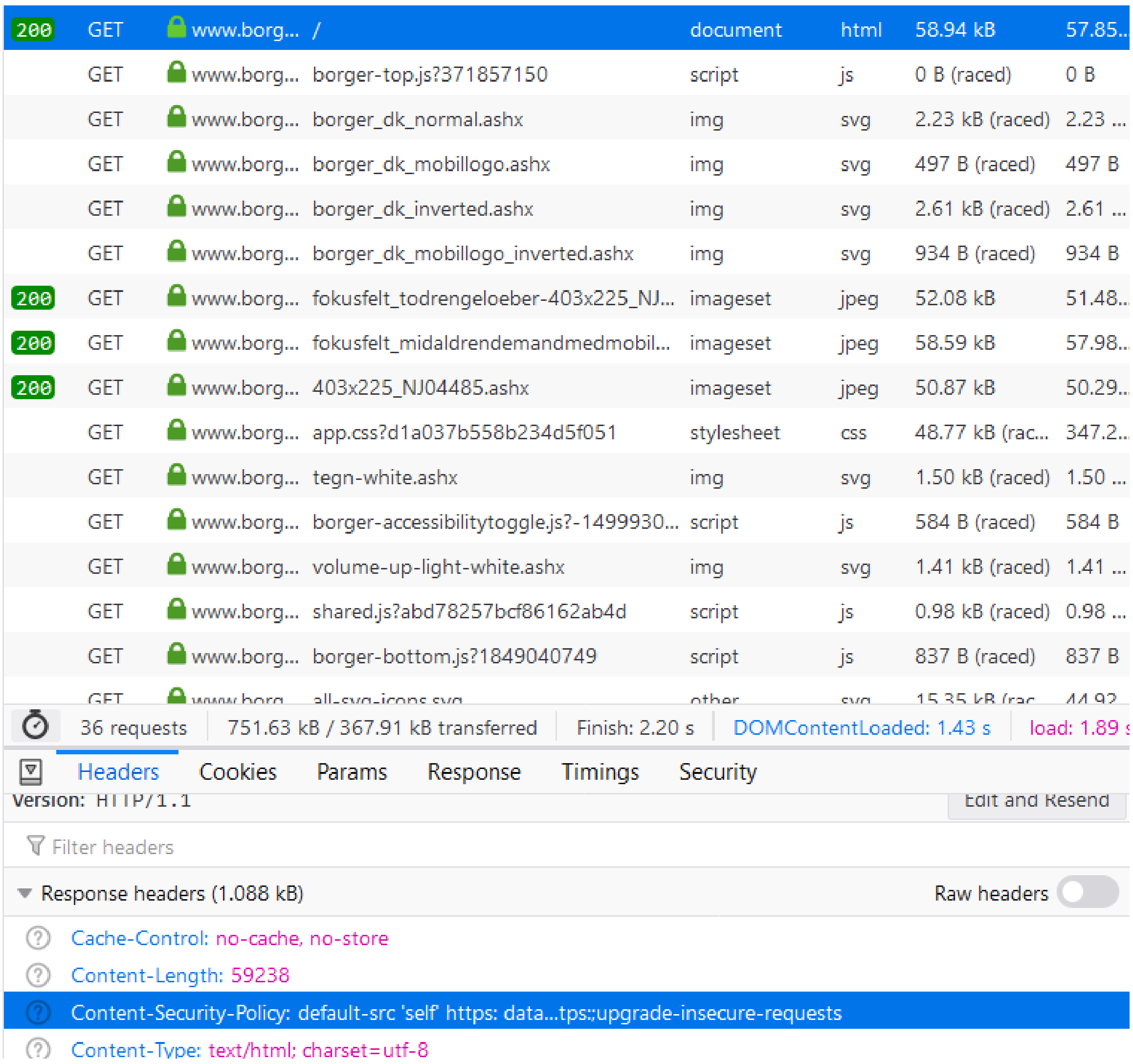
[16] West, M. and Medley, J. *Content Security Policy*. Google Developers. July 2017. URL: https://developers.google.com/web/fundamentals/security/csp/#if_you_absolutely_must_use_it_.

[17] *What is MIME Sniffing? - KeyCDN Support*. KeyCDN. URL: https://www.keycdn.com/support/what-is-mime-sniffing (visited on 11/15/2019).

A Tools used

- `whois` v5.5.2
- `nslookup`
- `nmap` v7.80
- `traceroute` v2.1.0
- OWASP ZAP 2.8.0

B CSP for `borger.dk`



200	GET	www.borg...	/	document	html	58.94 kB	57.85..
	GET	www.borg...	borger-top.js?371857150	script	js	0 B (raced)	0 B
	GET	www.borg...	borger_dk_normal.ashx	img	svg	2.23 kB (raced)	2.23 ...
	GET	www.borg...	borger_dk_mobillogo.ashx	img	svg	497 B (raced)	497 B
	GET	www.borg...	borger_dk_inverted.ashx	img	svg	2.61 kB (raced)	2.61 ...
	GET	www.borg...	borger_dk_mobillogo_inverted.ashx	img	svg	934 B (raced)	934 B
200	GET	www.borg...	fokusfelt_todrengeloeber-403x225_NJ...	imageset	jpeg	52.08 kB	51.48..
200	GET	www.borg...	fokusfelt_midaldrendemandmedmobil...	imageset	jpeg	58.59 kB	57.98..
200	GET	www.borg...	403x225_NJ04485.ashx	imageset	jpeg	50.87 kB	50.29..
	GET	www.borg...	app.css?d1a037b558b234d5f051	stylesheet	css	48.77 kB (rac...	347.2..
	GET	www.borg...	tegn-white.ashx	img	svg	1.50 kB (raced)	1.50 ...
	GET	www.borg...	borger-accessibilitytoggle.js?-1499930...	script	js	584 B (raced)	584 B
	GET	www.borg...	volume-up-light-white.ashx	img	svg	1.41 kB (raced)	1.41 ...
	GET	www.borg...	shared.js?abd78257bcf86162ab4d	script	js	0.98 kB (raced)	0.98 ...
	GET	www.borg...	borger-bottom.js?1849040749	script	js	837 B (raced)	837 B
	GET	www.borg...	all-svg-icons.svg	other	svg	15.35 kB (rac	44.92

36 requests | 751.63 kB / 367.91 kB transferred | Finish: 2.20 s | DOMContentLoaded: 1.43 s | load: 1.89 s

Headers Cookies Params Response Timings Security

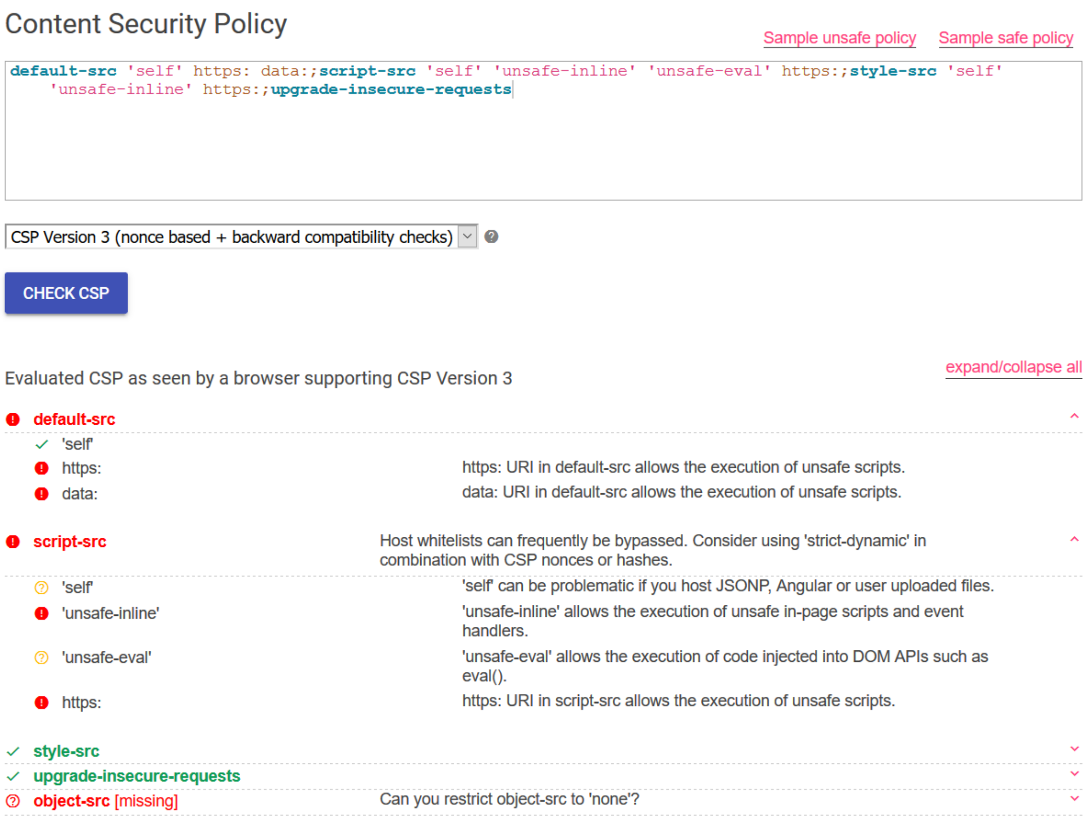
Version: HTTP/1.1 Edit and Resend

Filter headers

Response headers (1.088 kB) Raw headers

```
Cache-Control: no-cache, no-store
Content-Length: 59238
Content-Security-Policy: default-src 'self' https: data...tps:;upgrade-insecure-requests
Content-Type: text/html; charset=utf-8
```

Figure 9: Getting the CSP for borger.dk



```
default-src 'self' https: data:;script-src 'self' 'unsafe-inline' 'unsafe-eval' https:;style-src 'self'
    'unsafe-inline' https:;upgrade-insecure-requests
```

Evaluated CSP as seen by a browser supporting CSP Version 3

default-src	
✓ 'self'	
https:	https: URI in default-src allows the execution of unsafe scripts.
data:	data: URI in default-src allows the execution of unsafe scripts.
script-src	Host whitelists can frequently be bypassed. Consider using 'strict-dynamic' in combination with CSP nonces or hashes.
'self'	'self' can be problematic if you host JSONP, Angular or user uploaded files.
'unsafe-inline'	'unsafe-inline' allows the execution of unsafe in-page scripts and event handlers.
'unsafe-eval'	'unsafe-eval' allows the execution of code injected into DOM APIs such as eval().
https:	https: URI in script-src allows the execution of unsafe scripts.
✓ style-src	
✓ upgrade-insecure-requests	
object-src [missing]	Can you restrict object-src to 'none'?

Figure 10: Problems with the borger.dk CSP (according to [3])

C Keyboard Details

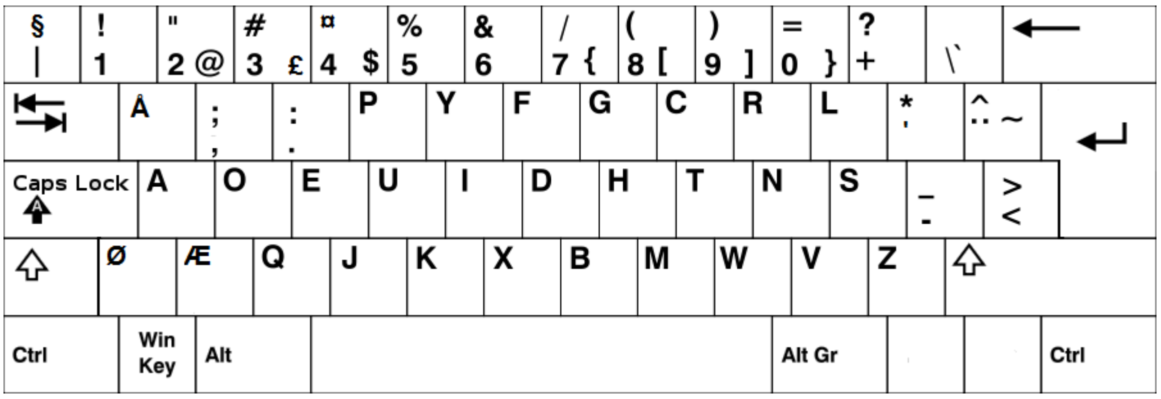
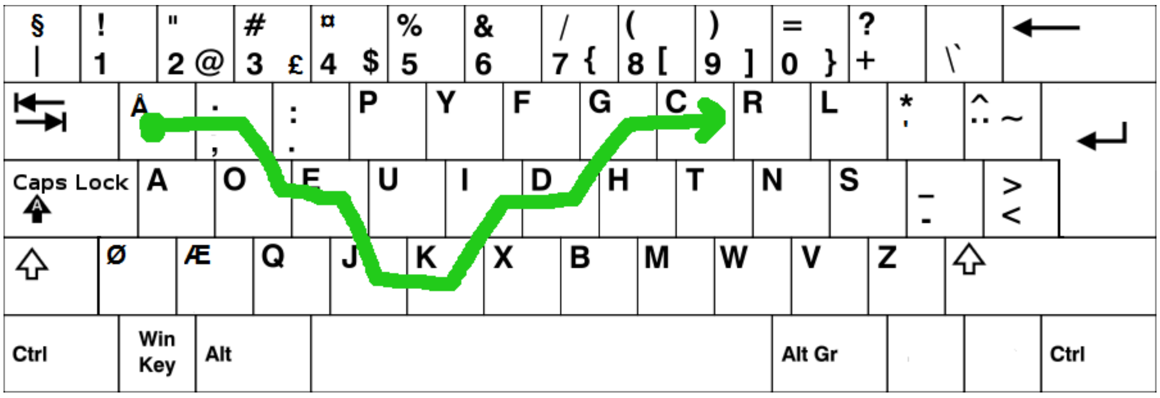
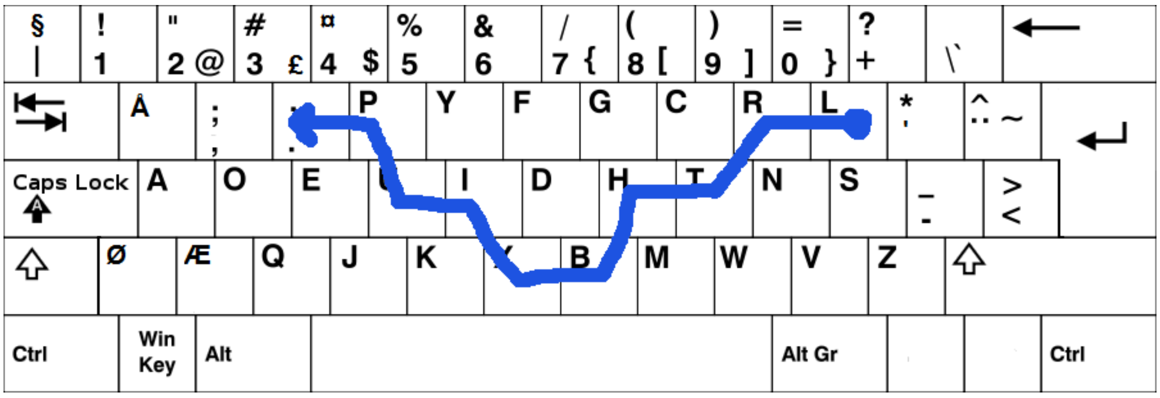


Figure 11: The Norwegian Dvorak keyboard layout



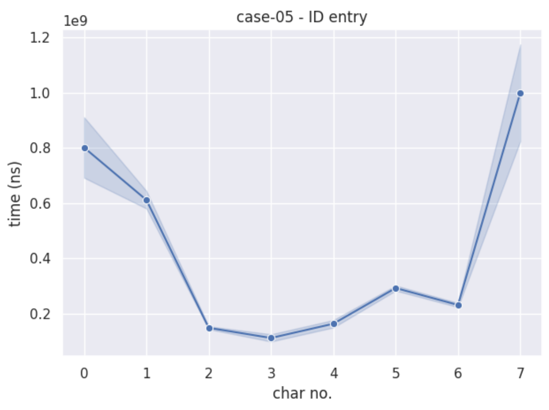
(a) Moving rigthwards using the right hand (case 9 pwd)



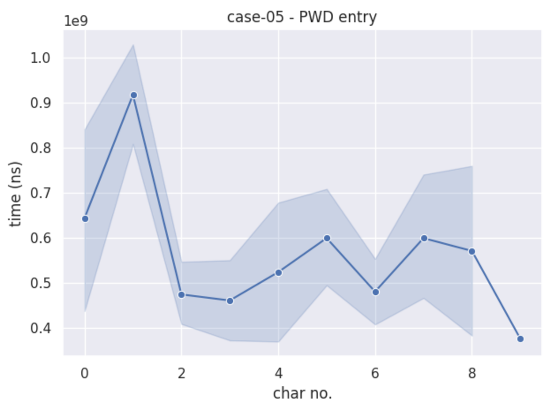
(b) Moving leftwards using the left hand (case 10 pwd)

Figure 12: Hand travel patterns for the pwd in cases 9 and 10

D Cases 5-8

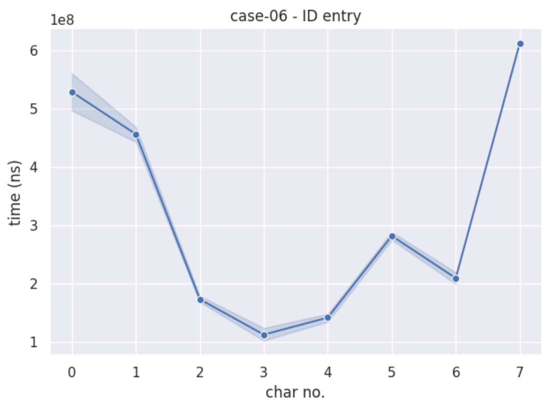


(a) Key timings for ID 5

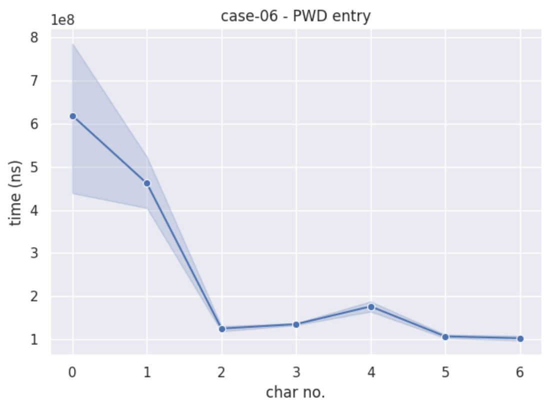


(b) Key timings for PWD 5

Figure 13: Timing results for case 5

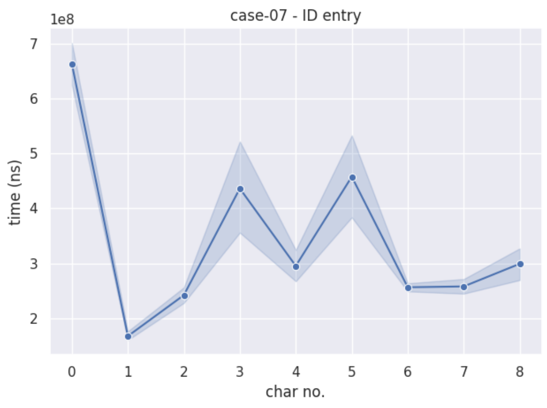


(a) Key timings for ID 6

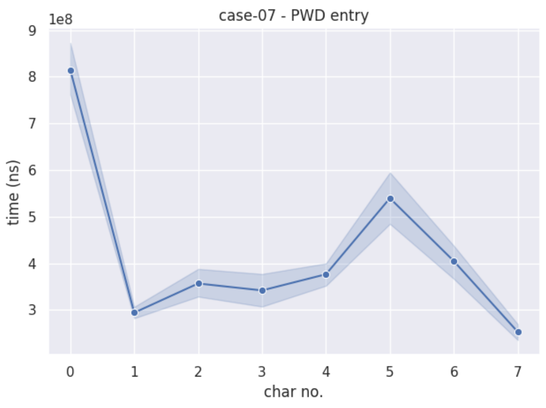


(b) Key timings for PWD 6

Figure 14: Timing results for case 6

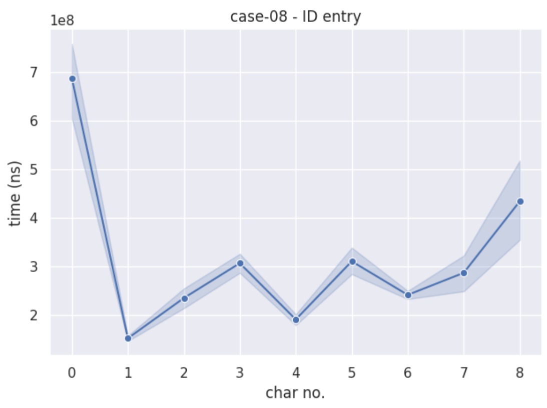


(a) Key timings for ID 7

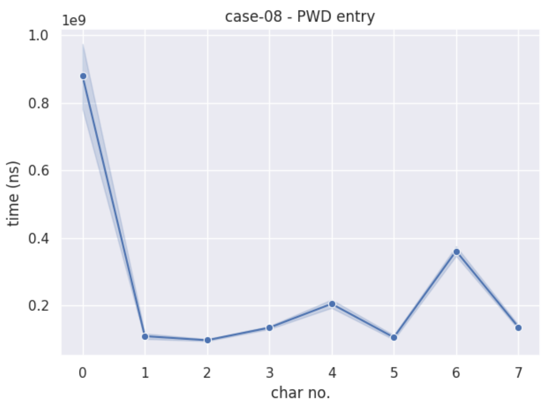


(b) Key timings for PWD 7

Figure 15: Timing results for case 7



(a) Key timings for ID 8



(b) Key timings for PWD 8

Figure 16: Timing results for case 8

E Code

E.1 keylogger.py

```
import csv
from pathlib import Path
import pynput.keyboard as kb
from time import time_ns

# query the user for a location to store the data
csv_file_path = input("Store the data in file: ")
if not csv_file_path.endswith('.csv'):
    csv_file_path += '.csv'

# initialise data fields as empty
travel_data = []
pattern_data = {}
id_pattern = []
pwd_pattern = []
prev_key = ''
# record ID and password in pairs
pwd = False


# path to keep sample count for that data
counter_file = csv_file_path[:-4] + '-counter'
if not Path(counter_file).exists():
    Path(counter_file).touch()
counter_file = open(counter_file, 'r+')
counter_file.seek(0)
try:
    sample_no = int(counter_file.readline())
except ValueError:
    sample_no = 0

# set up the CSV-writer for data storage
csv_file = open(csv_file_path, 'a+', newline='')
pattern_csv = csv.writer(csv_file, delimiter=',', quotechar='"')
if Path(csv_file_path).stat().st_size == 0:
    pattern_csv.writerow(['sample-no', 'input-type', 'char-no', 'time (ns)',
                          'total-time (ns)'])


def log(key):
    global prev_time
    global travel_data
    global id_pattern
    global pwd_pattern
    global pattern_data
    global prev_key
    global sample_no
    global pwd
    elapsed = time_ns() - prev_time
    if key == kb.Key.esc:
        # stop the keylogger
        return False
    elif key != kb.Key.enter:
        travel_data.append((prev_key, str(key), elapsed))
        if pwd:
            # store intervals as password
            pwd_pattern.append(elapsed)
        else:
            # store intervals as user ID
            id_pattern.append(elapsed)
    elif not pwd:
        # if we just got the username, get the password
        pwd = True
    else:
        # record the data
        id_sum = sum(id_pattern)
        pwd_sum = sum(pwd_pattern)
        for i in range(len(id_pattern)):
            pattern_csv.writerow([sample_no, 'id', i, id_pattern[i], id_sum])
        for i in range(len(pwd_pattern)):
            pattern_csv.writerow([sample_no, 'pwd', i, pwd_pattern[i], pwd_sum])
        sample_no += 1
        print('\n', sample_no)
        counter_file.seek(0)
        counter_file.write(str(sample_no) + '\n')
        id_pattern = []
        pwd_pattern = []
        pwd = False
    # print(key, elapsed)
    prev_time = time_ns()
    prev_key = str(key)


# start the keylogger
with kb.Listener(on_press=log) as kb_listener:
    prev_time = time_ns()
    print(sample_no)
    kb_listener.join()
```

E.2 plot.py

```
import argparse

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

parser = argparse.ArgumentParser()
parser.add_argument('csv_file', help='The csv-file to plot')
parser.add_argument('--sem', action='store_true',
                    help='aggregate the samples and plot SEM along with the average')
args = parser.parse_args()

# enable seaborn for pretty plots
sns.set(style='darkgrid')

# read the file
if args.sem:
    df = pd.read_csv(args.csv_file)
    # filter based on input-type
    id_df = df.loc[df['input-type'] == 'id']
    pwd_df = df.loc[df['input-type'] == 'pwd']
    # plot the dfs with standard error (ci=68)
    id_plt = sns.lineplot(x='char-no', y='time (ns)', data=id_df, ci=68,
                          marker='o')
    plt.figure()
    pwd_plt = sns.lineplot(x='char-no', y='time (ns)', data=pwd_df, ci=68,
                           marker='o')
else:
    df = pd.read_csv(args.csv_file, index_col=[0, 1, 2])
    # transform the multi-index to get time taken; dropna fixes the case where
    # len(id) != len(pwd)
    id_df = df.unstack(level=1)['time (ns)', 'id'].unstack(level=0).dropna()
    pwd_df = df.unstack(level=1)['time (ns)', 'pwd'].unstack(level=0).dropna()
    # plot the dfs
    id_plt = id_df.plot()
    pwd_plt = pwd_df.plot()

# configure title and labels
id_plt.set_title(args.csv_file[:-4] + ' - ID entry')
id_plt.set_xlabel('char no.')
id_plt.set_ylabel('time (ns)')
pwd_plt.set_title(args.csv_file[:-4] + ' - PWD entry')
pwd_plt.set_xlabel('char no.')
pwd_plt.set_ylabel('time (ns)')
plt.show()
```