

Sunday, 14 April 2019

Vulnerability Report

Replay Of Persistent Authentication Token in HTTP GET Request URL Path

IMPACT: CRITICAL

Affected Software:

SimplyBook.me / SimplyBook.it
Online Booking and Scheduling
System for Service Based Industries

Affected Users:

Management Portal & Client Accounts
Are Affected

Breaching:

**CONFIDENTIALITY, INTEGRITY &
ACCESSABILITY** of User Data

URLs used for testing:

<https://cybrgradeuk.simplybook.it/>

<https://cybrgradeuk.secure.simplybook.it/>

Details of tester:

Stu Patterson
CybrGrade UK
contact@cybrgrade.com



TABLE OF CONTENTS

TABLE OF CONTENTS	1
REPLAY OF PERSISTENT AUTHENTICATION TOKEN IN HTTP GET REQUEST URL PATH.....	2
EXPLOIT EXAMPLES	4
REFERENCES.....	8

REPLAY OF PERSISTENT AUTHENTICATION TOKEN IN HTTP GET REQUEST URL PATH

The Vulnerability

When any [SimplyBook.it/SimplyBook.me](#) management portal administrator/user or booking/scheduling service provider client account is created, an immutable MD5 hash value is generated and associated with that account in perpetuity. The purpose of the hash is to allow a 1-click login and password reset function for **administrators, users, & clients** of both the services booking/scheduling website and the services management portal.

The **login-link** URL contains the subdomain for a particular booking site hosted on [simplybook.it](#) and a token in the form an **MD5 hash** value that appears to be **created the first time it is used and stored in the database for the duration of the life of the account**. This hash represents a fully authenticated session for the associated user, and anyone who has knowledge of the hash has the ability to **perpetually re-authenticate as that user regardless of any password, username, email address changes made to the details of that account**.

Because the hash does not expire and it is passed as part of the URL in a GET request, this means it will be recorded in both browser history logs, and business web monitoring service logs. So if a user accesses their account using a **login-link** via a shared system and even makes sure to log out and clear their session cookies afterwards, the full URL path is still recorded in the browser history and anyone with access to the browser can access and use it to access the account associated with that **login-link**.

Clients of the booking service website

When the “**Client login**” custom feature is enabled on a [SimplyBook.me](#) site, both the send login link and password reset features send a “**login-link**” hyperlink via email to the client’s registered email account when activated. For client accounts **login-link** is in the form of:

[https://\[SITENAME\].simplybook.it/v2/client/restore-password/hash/\[MD5-HASH\]](https://[SITENAME].simplybook.it/v2/client/restore-password/hash/[MD5-HASH])

User accounts used for accessing the management portal

Management portal accounts have a **login-link** in the form of:

[https://\[SITENAME\].secure.simplybook.it/login/by-hash/code/\[MD5-HAS\]?reset-password=true](https://[SITENAME].secure.simplybook.it/login/by-hash/code/[MD5-HAS]?reset-password=true)

Other points of notes: Also, the GET parameter of “**reset-password=true**” is irrelevant and does not force the password to be changed if the user clicks away from the password change prompt in the GUI.

Additionally, when an administrator resets a password for a user account, or resends the **login-link** via email to another management portal user account, an additional hash is generated but all previous hashes still remain valid and do not expire. In testing I was able to generate 164 separate MD5 Hashes for a single user account and none of them were ever observed to expire until the account itself was deleted.

Observations

Tokens used for password reset and account recovery should be **randomly generated** and be **single use** only. They should expire as soon as either one of the following conditions are met:

1. After a specified short amount of time (e.g. 5 mins)
2. After they are used.

Account owners should be sent a notification email upon creation and use of a **login-link** and also when any changes are made to their user details or account information. (e.g change of email address or phone number).

Due to time limitations I have not fully carried out entropy testing or reverse engineering of the MD5 generation algorithm in this case. However, if a deterministic dataset (e.g a combination of account details + time or + static salt value) is being used to generate the MD5 hash values this leaves the system fully open to remote compromise of accounts and services hosted on the platform because this approach is open to reverse engineering and **login-link** prediction attacks resistant to account lockout due to the system design.

Because [SimplyBook.me](https://simplybook.me) have implemented user data access and handling features to enable compliance with the **European General Data Protection Regulation (GDPR)**, if service providers enable the feature on their site, this additionally risks detailed and complete repositories of user data becoming accessible to attackers. This is because, anyone with a valid **login-link** for a client account is able to reconfigure the email address for that account, get an access code via their own email, download a full copy of the account owners **Personal Data Report**, then change the email back to the original without the account owner being made aware. This scenario is demonstrated below and would lead to a full breach of **CONFIDENTIALITY, INTEGRITY** and **ACCESSIBILITY** of client account data.

ref: <https://simplybook.me/en/gdpr-compliance>

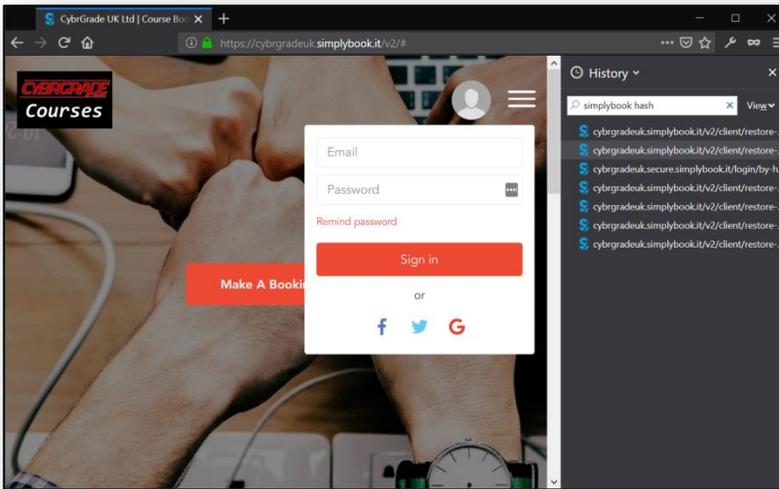
For security reasons it is becoming common practice within secured business networks for web traffic to be monitored, and for GET requests originating from within their networks to be logged. A typical sample of a logged GET request will contain the full URL path. As you can see **login-link** values would also be logged in such a situation also.

```
root@CybrGradeUK:~# grep restore-password /var/log/nginx/access.log
162.158.155.58 - - [12/Apr/2019:12:49:11 +0000] "GET /v2/client/restore-password
/hash/3226bcffca588b692357b2f4a9204a86 HTTP/1.1" 302 97937 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.368
3.103 Safari/537.36"
root@CybrGradeUK:~#
```

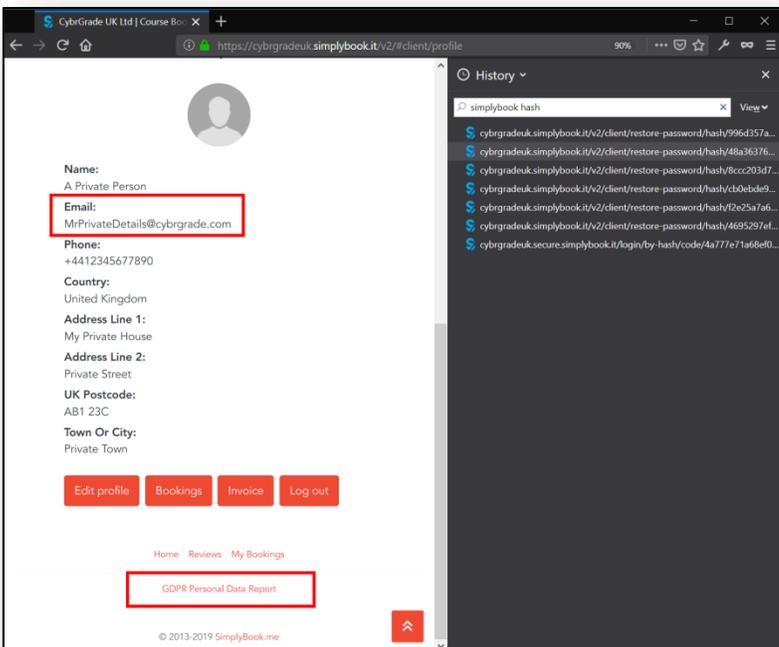
EXPLOIT EXAMPLES

Here I will demonstrate how an attacker that finds a **login-link** in the history of the browser they have shared access to can lead to full loss on **CONFIDENTIALITY, INTEGRITY & ACCESSABILITY** of the data associated with account the **login-link** is associated with.

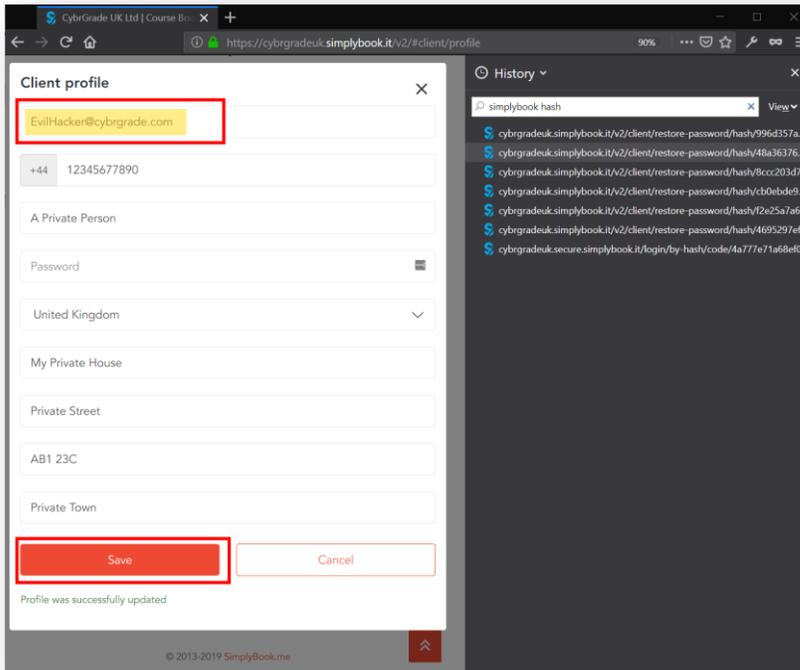
1. Starting without a user session, we can open the history search sidebar and search for “**simplybook hash**” and get a list of all the **login-link** URLs saved in the browser history.



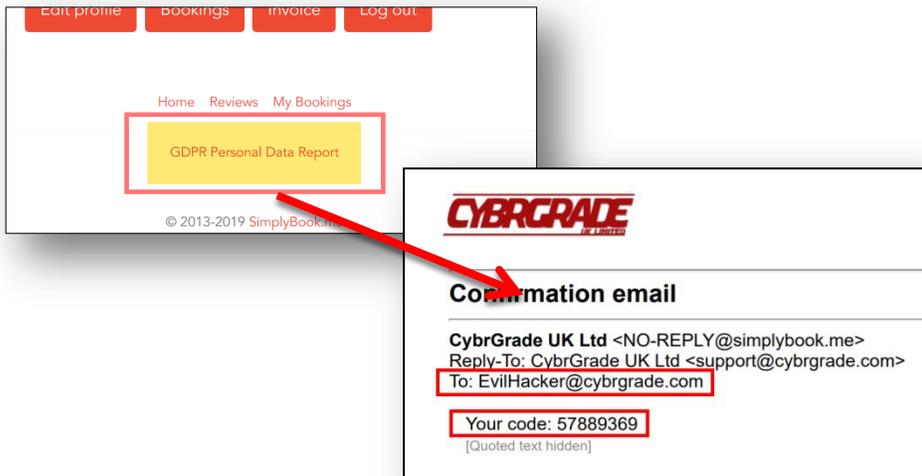
2. Next, by opening one of them we are instantly granted access to the associated user account. Even though the link has been used previously, the user logged out, and no session state was saved in the browser.



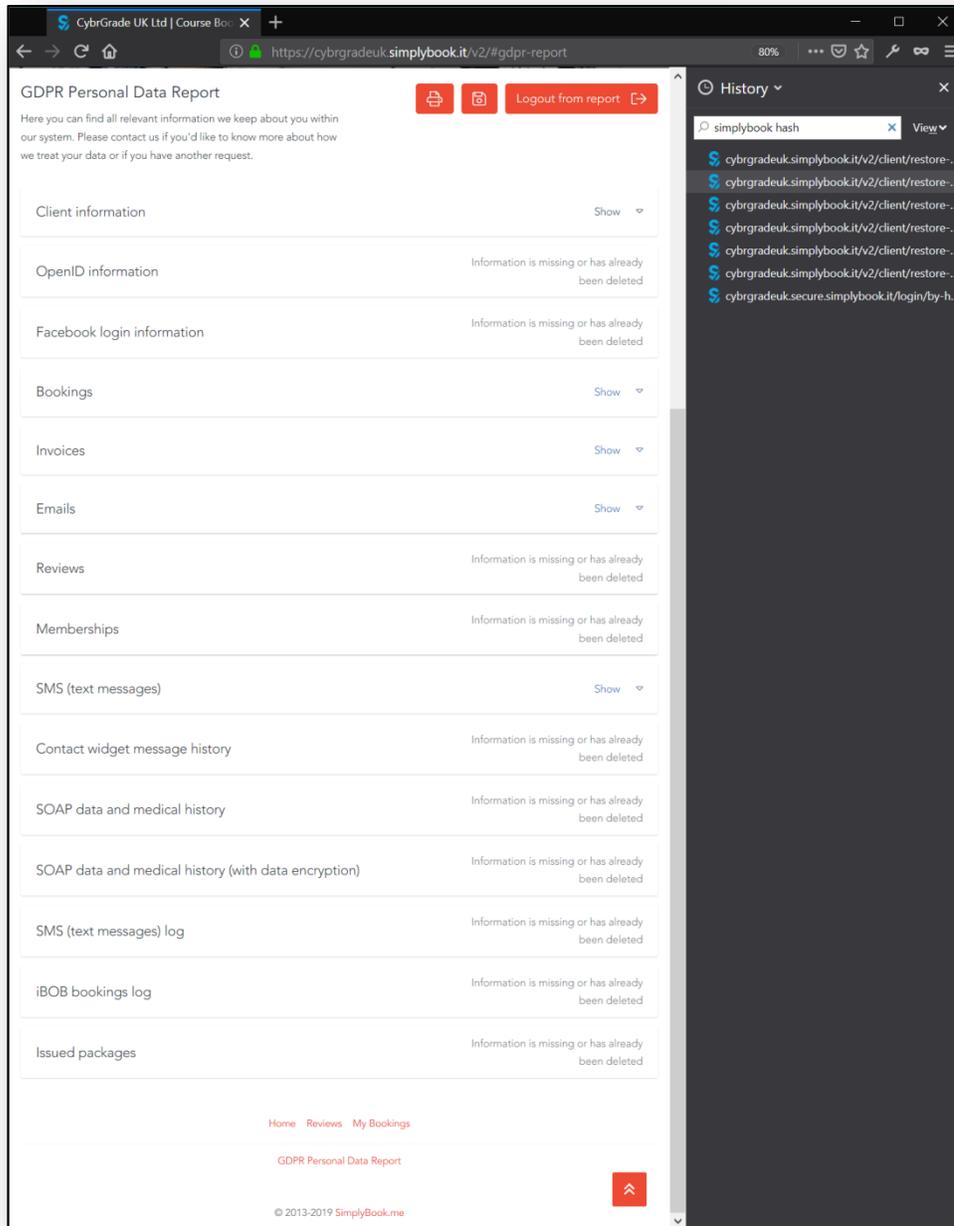
3. Now an attacker is able to change the account email address to one that they control.



4. Then they can safely click to read the **Personal Data Report** of the owner of the account and intercept the temporary access code without the data owner being notified.

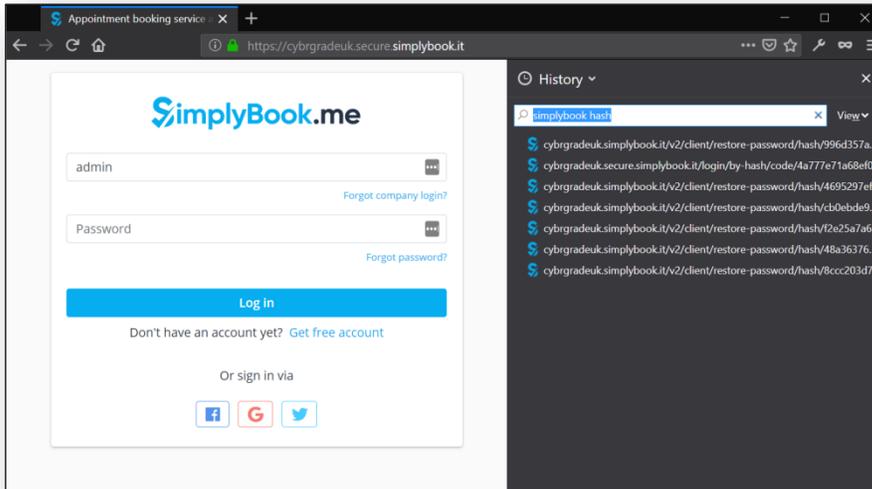


5. The attacker can now export all data ever generated by or on behalf of that associated account identity.

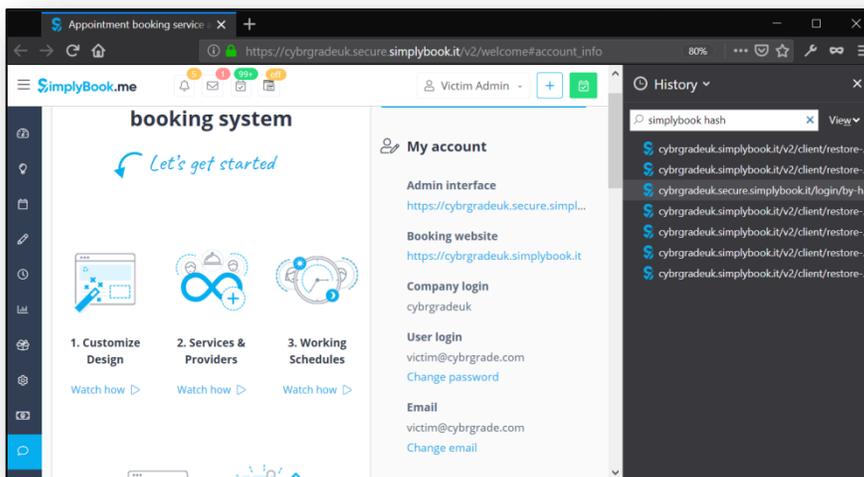


The same kind of attack is also possible for any management portal accounts and would lead to an even more devastating attack if combined with the **escalation of privilege attack** demonstrated in my previous report ([Report SimplyBookIt Privesc by CybrGradeUKLtd.pdf](#)).

1. Starting off without any session, an attacker can use a **login-link** from the browser history to access the management portal site as the associated user. ([SITENAME].secure.simplybook.it)



2. Then by simply using that saves link we instantly become an administrator.



REFERENCES

OWASP Good Practice Cheat Sheets

https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/REST_Security_Cheat_Sheet.md
https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Authentication_Cheat_Sheet.md

Site Management Portal Privilege Escalation

https://drive.google.com/file/d/1Oc0lp6hpsyoKiBjMHVdCJDHUrqlpWM_T/view
password: 6oOm4f6RfLqRkq0vn7UGjOZmfgrFFsf1