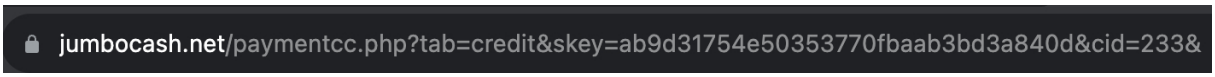
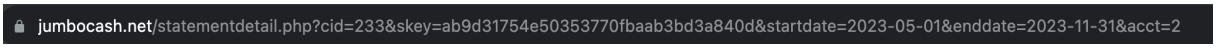
Where I started

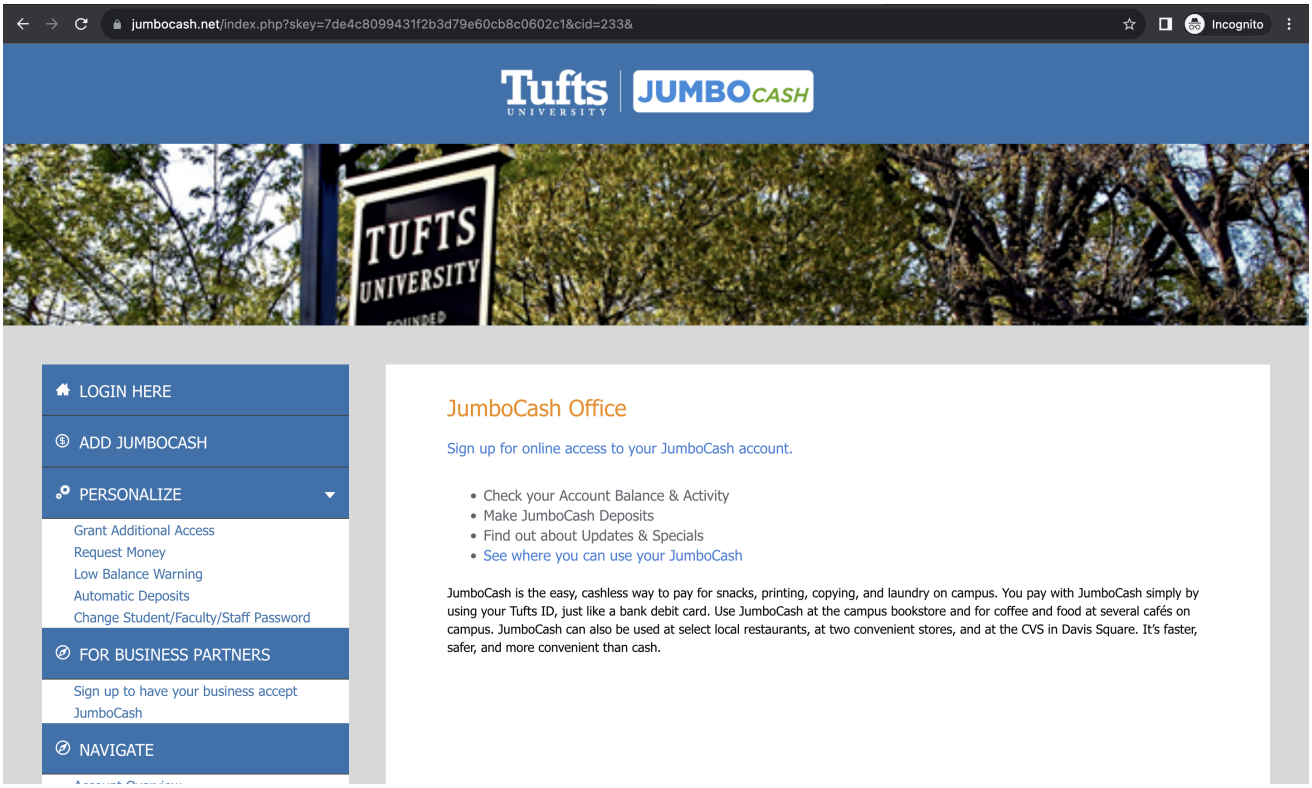
I was sitting in the library with my friend and we wanted to order food from "Commons Marketplace". I didn't have enough JumboCase on my account, so I went to the JumboCash page to add funds. Upon logging into the website and browsing to the payment page, I looked at the URL and saw smth like this:



Curious, I decided to head on over to where SPII is stored, in the statements. There I saw a URL similar to this:



From the URL you can pinpoint exactly what the *skey* is: **ab9d31754e50353770fbaab3bd3a840d&** I thought "Surely this isn't a static key that is used for every account upon login, right?" I copied my entire URL and pasted into an incognito page, BOOM, everything was there. Palms sweating, I asked my friend to send me their URL upon logging onto JumboCash and this was her *skey*: **7de4c8099431f2b3d79e60cb8c0602c1&**. Scared, I replaced my *skey* with hers and got this:



"Dang it". I tried the same for the statement page "statement.php" and got a **NOI** blank page. After trying some basic SQL injection commands such as *' OR '1'='1* to no avail, I remembered using sqlmap during our CTF. I raced to our CTF writeup and found the section on SQL Injection using sqlmap. I ran sqlmap and after a few minutes of hitting "yes" to testing a certain .php page, I tried accessing JumboCash and got hit with this:

Your IP was locked due to too many request. Please try again after 5 mins

Then, I looked back at my terminal and saw that after each test, SQLMap sent me this warning:

```
[12:37:12] [ERROR] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to
 perform more tests. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--
tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent', skipping to the next target
[12:37:12] [WARNING] HTTP error codes detected during run:
429 (Too Many Requests) - 147 times
```

Soooo, everything I had been running for the past few min was probably all useless because my IP had been blocked...

Wrap Up Thoughts

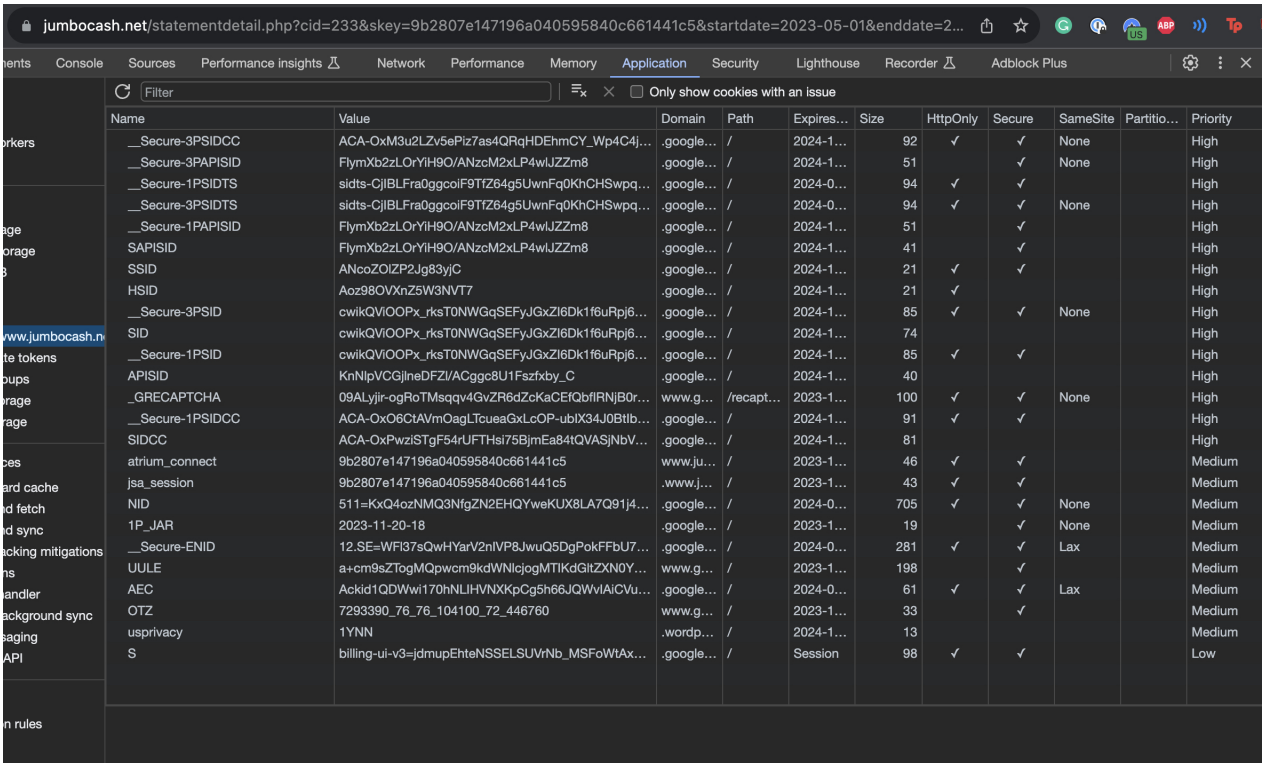
So, after a rollercoaster of emotions, I came to this conclusion: Good job Tufts Security Team. JumboCash was protected against SQLmap scripts as it times out IPs that send too many requests to the server. As for the *skey* situation, playing around with it further led me to these observations:

- The *skey* is generated upon starting the user's session (logging into the JumboCash website)
- The *skey* is somehow protected/authenticated by the browser's cookie (Can still "login" to JumboCash page on the same browser in incognito mode, but not on a different browser or computer)

I was shocked to see how every php page uses the *skey* to validate the user's permission to view "x" page and how everything was just given/visible in the URL bar. I was especially shocked at how the website was coded as modifying the "Start/End" Dates on the statement page actually changes what is shown. Attempting to sql inject the date results in an "invalid date" message. The *cid=233&* is the same for everyone and modifying that results in an error page.

Future Steps

So, although the JumboCash website is coded with "exposed" security information, it has measures behind it to protect the user and whatever SPII/PII is stored in its databases. However, even though it has these security measures, having exposed security keys isn't a good practice and who knows there is some way to exploit the communication between the cookie and the skey authentication. This is what I found upon looking at the Cookies during my active session:



Name	Value	Domain	Path	Expires...	Size	HttpOnly	Secure	SameSite	Partitio...	Priority
__Secure-3PSIDCC	ACA-OxM3u2LZv5ePiz7as4QRqHDEhmCY_Wp4C4j...	.google...	/	2024-1...	92	✓	✓	None		High
__Secure-3PAPISID	FlymXb2zLOrYiH9O/ANzcM2xLP4wlJZZm8	.google...	/	2024-1...	51		✓	None		High
__Secure-1PSIDTS	sidts-CjIBLFra0ggcoiF9TfZ64g5UwnFq0KhCHSwpq...	.google...	/	2024-0...	94	✓	✓			High
__Secure-3PSIDTS	sidts-CjIBLFra0ggcoiF9TfZ64g5UwnFq0KhCHSwpq...	.google...	/	2024-0...	94	✓	✓	None		High
__Secure-1PAPISID	FlymXb2zLOrYiH9O/ANzcM2xLP4wlJZZm8	.google...	/	2024-1...	51		✓			High
SAPISID	FlymXb2zLOrYiH9O/ANzcM2xLP4wlJZZm8	.google...	/	2024-1...	41		✓			High
SSID	ANcoZOlZP2Jg83yjC	.google...	/	2024-1...	21	✓	✓			High
HSID	Aoz98OVXnZ5W3NVT7	.google...	/	2024-1...	21	✓				High
__Secure-3PSID	cwikQViOOPx_rksT0NWGqSEFyJGxZI6Dk1f6uRpj6...	.google...	/	2024-1...	85	✓	✓	None		High
SID	cwikQViOOPx_rksT0NWGqSEFyJGxZI6Dk1f6uRpj6...	.google...	/	2024-1...	74					High
__Secure-1PSID	cwikQViOOPx_rksT0NWGqSEFyJGxZI6Dk1f6uRpj6...	.google...	/	2024-1...	85	✓	✓			High
APISID	KnNlpVCGjlneDFZl/ACggc8U1Fszfxby_C	.google...	/	2024-1...	40					High
_GRECAPTCHA	09ALyjir-ogRoTMsqqv4GvZR6dZcKaCEfQbfIRNjB0r...	www.g...	/recapt...	2023-1...	100	✓	✓	None		High
__Secure-1PSIDCC	ACA-OxO6CtAVmOagLTcueaGxLcOP-ubIX34J0BtIb...	.google...	/	2024-1...	91	✓	✓			High
SIDCC	ACA-OxPwziSTgF54rUFTHsi75BjmEa84tQVASjNbV...	.google...	/	2024-1...	81					High
atrium_connect	9b2807e147196a040595840c661441c5	www.ju...	/	2023-1...	46	✓	✓			Medium
jsa_session	9b2807e147196a040595840c661441c5	.www.j...	/	2023-1...	43	✓	✓			Medium
NID	511=KxQ4ozNMQ3NfgZN2EHQYweKUX8LA7Q91j4...	.google...	/	2024-0...	705	✓	✓	None		Medium
1P_JAR	2023-11-20-18	.google...	/	2023-1...	19		✓	None		Medium
__Secure-ENID	12.SE=WFl37sQwHYarV2nIVP8JwuQ5DgPokFFbU7...	.google...	/	2024-0...	281	✓	✓	Lax		Medium
UULE	a+cm9sZTogMQpwcm9kdWNlcjogMTlKdGltZXN0Y...	www.g...	/	2023-1...	198		✓			Medium
AEC	Ackid1QDWwi170hNLIHVNXKpCg5h66JQWvIAiCVu...	.google...	/	2024-0...	61	✓	✓	Lax		Medium
OTZ	7293390_76_76_104100_72_446760	www.g...	/	2023-1...	33		✓			Medium
usprivacy	1YNN	.wordp...	/	2024-1...	13					Medium
S	billing-ui-v3=jdmupEhteNSSELSUVrNb_MSFoWtAx...	.google...	/	Session	98	✓	✓			Low

The current *skey* can be found in the "atrium_connect" and "jsa_session" sections. I tried changing those values for a new key and replacing the URL skey, but got "logged out". I'm hesitant to say that this dynamic key security is foolproof, but as of right now, there haven't been any signs of easy exploitation methods.