Silent Signals

Exploiting Security and Privacy Side-Channels in End-to-End Encrypted Messengers

Motivation

DEFCON

Motivation

- WhatsApp is world's largest messenger
  - Used for private and professional communication
  - Over 3 billion users

- Signal is known for its focus on privacy
  - Also used by some government officials ;)
  - About 70 million users

- Anybody using these apps is affected by our findings
  - Only requirement is having a person's phone number

DEFCON

The Speakers

Gabriel:

- PhD candidate at University of Vienna
- Security researcher at SBA Research
- Cellular networks and mobile security
- Speaker at multiple conferences

Max:

- Bachelor’s thesis supervised by Gabriel
- Master’s student at University of Vienna
- Bug bounty hunter
- First time speaker

DEFCON

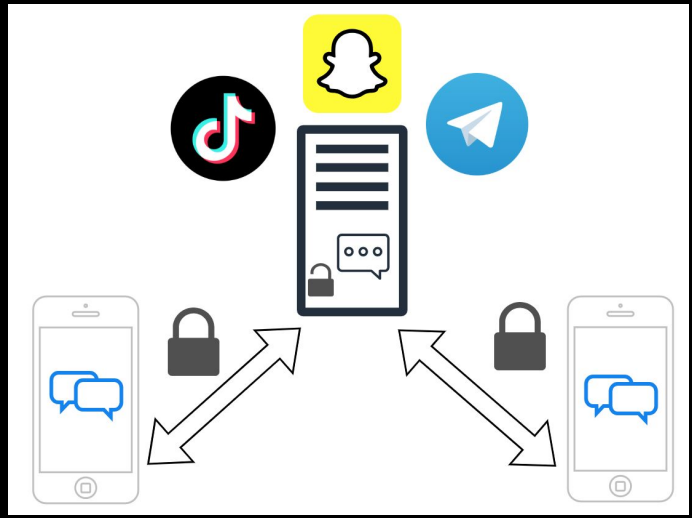
The Story

- Started as Max’s bachelor thesis
- Ended up discovering vulnerabilities affecting billions of users
- Small team of independent researchers
- Everyone can still find flaws in refined systems

Background

Encryption in Instant Messengers

- Before End-to-End Encryption (E2EE)
  - Transport encryption: messages between the user device and the messaging service are encrypted (e.g. via TLS)
    - Protection against wiretapping
  - However: messaging service can read the clear text message
  - Examples: TikTok, Snapchat, Telegram



DEFCON

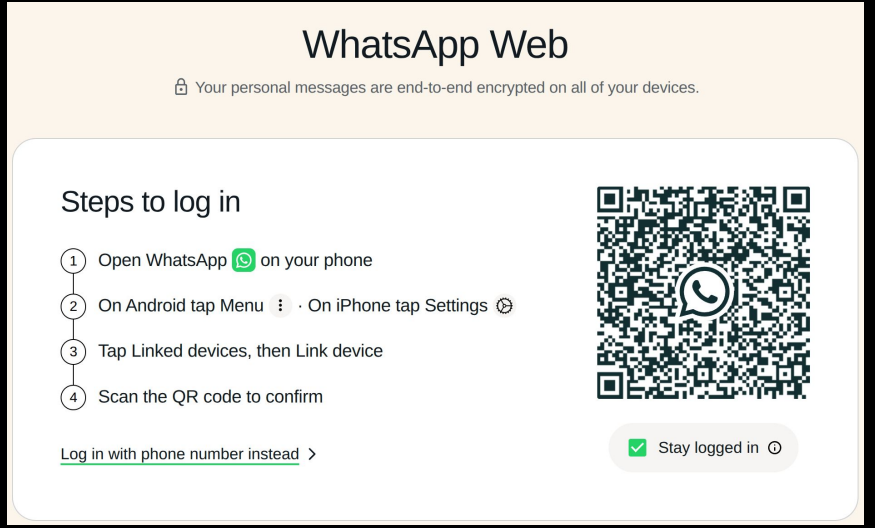
End-To-End Encryption (E2EE)

- End-to-End Encryption (E2EE)
  - Messages are encrypted between two communication parties
    - Every user device has their own keys
  - Messaging service is not able to inspect messages
    - Breaking lawful interception
  - Examples: Signal, WhatsApp, Messenger

Multi-Device Setups with E2EE

- Most messengers allow adding additional *companion devices*
  - Desktop or Web Application (e.g., WhatsApp Web)

- Two approaches:
  - Leader-based approach
  - Client fanout



DEFCON

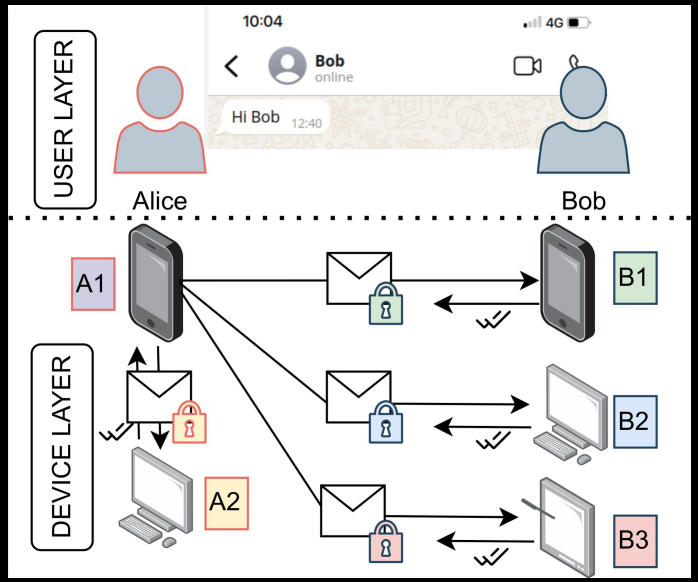
Multi-Device Setups with E2EE

- Leader-based approach
  - Leader device (smartphone) is handling external message delivery
  - Internal devices synchronize with main device
  - Main device is single point of failure
    - When offline, no messaging possible
  - Used by WhatsApp until 2021

https://engineering.fb.com/2021/07/14/security/whatsapp-multi-device/

Multi-Device Setups with E2EE

- Client-Fanout
  - Actual E2EE between all devices
    - Every device has its own keys
  - Messages are encrypted individually for all target devices
    - Individually sent to all devices of the recipient
    - Also mirrored to own companion devices
  - Used by WhatsApp, Signal



DEFCON

Addressing in Multi-Device Setups

- Every phone number can have multiple devices
  - Addressed using device indexes
    - Main device has index 0

- Privacy leaks
  - Active device list can be consistently monitored
  - Due to the E2EE principle, any received message also leaks the device that was used to send it
    - Could already disclose the physical location of the sender

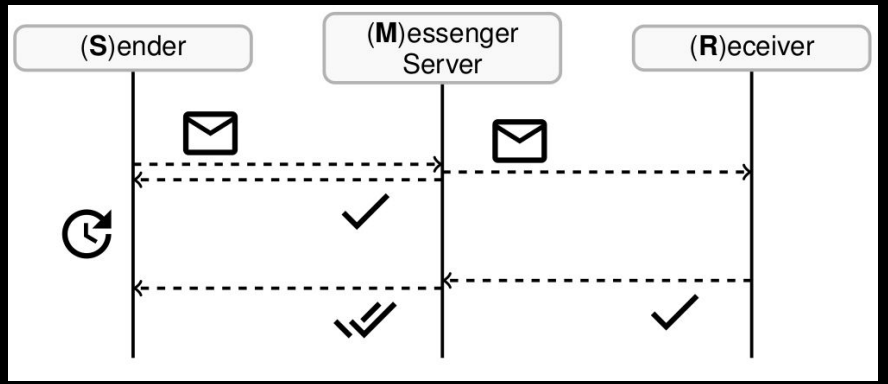
```
pogo@prekey:~$ ./query-devices -t 123456789
Querying registered devices for target number.

Found 3 existing devices: [0, 1, 3]
```

Message Delivery Process

- Delivery- and read receipts
  - Convenience features for message sender
  - However: delivery receipt also used as a control flow message
    - Confirms successful decryption of the message

✓ message is at server
✓✓ message is at receiver
✓✓ message was read



Schnitzler et al., Hope of Delivery: Extracting User Locations From Mobile Instant Messengers

DEFCON

Crash Course: Side-Channel

- Every action has (sometimes unwanted) side effects
  - Can be observed by third parties
  - Accidental information leak

- Real Life Examples
  - Less power consumption when not at home (e.g., holiday)

- Side channels in instant messengers
  - Delivery receipts and read receipts
  - Online Status

Vulnerabilities

DEFCON

Issue 1:

Delivery Receipt-based Timing Side Channel

Careless Whisper: Exploiting Silent Delivery Receipts to Monitor Users on Mobile Instant Messengers

Gabriel K. Gegenhuber[1], Maximilian Günther[1], Markus Maier[1], Aljosha Judmayer[1], Florian Holzbauer[1], Philipp É. Frenzel[2], and Johanna Ullrich[1]

[1]University of Vienna, [2]SBA Research

***Abstract*—**
With over 3 billion users globally, mobile instant messaging apps have become indispensable for both personal and professional communication. Besides plain messaging, many services implement additional features such as delivery and read receipts informing a user when a message has successfully reached its target. This paper highlights that delivery receipts can pose significant privacy risks to users. We use specifically crafted messages that trigger silent delivery receipts allowing any user to be pinged without their knowledge or consent. By using this technique at high frequency, we demonstrate how an attacker could extract private information such as following a user across different companion devices, inferring their daily schedule, or deducing current activities. Moreover, we can infer the number of currently active user sessions (i.e., main and companion devices) and their operating system, as well as launch resource exhaustion attacks, such as draining a user's battery or data allowance, all without generating any notification on the target side. Due to the widespread adoption of vulnerable messengers (*WhatsApp* and *Signal*) and the fact that any user can be targeted simply by knowing their phone number, we argue for a design change to address this issue.

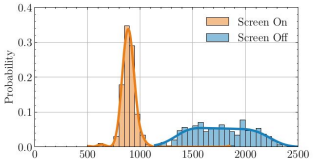


Fig. 1: Round-trip times (RTT) of delivery receipts, which are ≤ 1 second for *Screen On* states and > 1 second and above for *Screen Off* states measured on an iPhone using WhatsApp with a sampling rate of 1 Hz.

I. Introduction

Instant messengers serve a vast global audience with WhatsApp alone reaching over 2 billion users [22] and handling billions of messages daily. Besides being very common in general, instant messaging services are also used by high-profile government officials for sensitive conversations [1], [8], which adds an entirely different dimension to privacy issues within these services. In this paper, we present a novel privacy and availability attack vector on instant messaging systems, leveraging the (ab)use of *delivery receipts*.

There are two basic ways used by instant messengers to inform senders about message delivery, namely *delivery receipts* (consisting of *server ack & device ack)* and *read receipts*. The first acknowledges a message's receipt at the server or the destination device, the latter their view by the destination device's user. Read receipts have been misused to spy on conversation partners [7] and nowadays messenger applications allow to disable them in their privacy settings. Delivery receipts, however, cannot be deactivated due to design choices of the underlying protocol. Previous work has triggered delivery receipts through sending regular text messages in ongoing conversations and thereby showed that, based on the measured round-trip times (RTTs), country-level geolocation of a user's device is feasible [16]. Regular messages, however, trigger notifications for the target user, potentially alerting them to the ongoing attack, particularly when the probing messages are sent frequently.

Using techniques described in this paper, an adversary can craft stealthy messages that enable probing a target at high frequency (up to sub-second granularity) while not causing any notification at the target side and also in the absence of an ongoing conversation. With such an increased sampling rate, we systematically show that delivery receipts can leak user information beyond the country level.

For example, we show that the on/off state of a mobile phone's screen manifests in the delivery receipts' timing, see Figure 1, and, among others, allows to track the victim's screen time.

Moreover, we demonstrate that a user's activity can be followed across multiple devices (i.e., smartphone and companion sessions), creating further (and more severe) monitoring and tracking possibilities. In addition to utilizing delivery receipt RTT as a timing side channel, we demonstrate that implementation inconsistencies across different target architectures also leak information about the operating systems and application

DEFCON

Testing Methodology

- Experimenting with official clients
- Using third-party client implementations to
  - Test different message types (edit, react, delete)
  - Craft malformed messages

WhatsApp Clients: whatsmeow, Baileys, Cobalt
Signal Clients: signal-cli

Delivery Receipt Side-Channel

- In multi-device settings, delivery receipts are independently issued by each device
  - Can be used to ping each target device individually
  - Delivery Receipts also issued for message reactions

Action	Delivery Receipt			Push Notification		
Message	●	●	●	●	●	●
Reaction	●	●	○	◐	◐	○
Edit	●	●	○		○	○
Delete	●	●	○	○	○	○

 Edits cause (silent) notifications for iOS users only (no notifications are shown on Android).

DEFCON

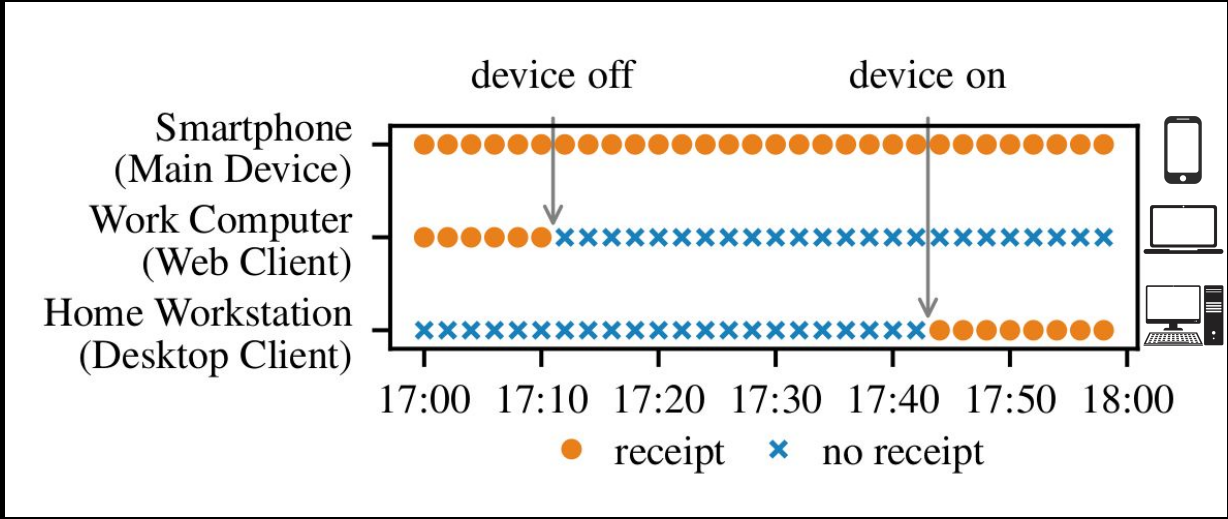
Silent Pings via Delivery Receipts

- Specifically crafted messages trigger a delivery receipt, but are not shown on the target's device
  - Reacting to an non-existent message ID

- Allowing an attacker to silently ping target devices
  - Pings can be sent repeatedly over long periods (e.g., days)

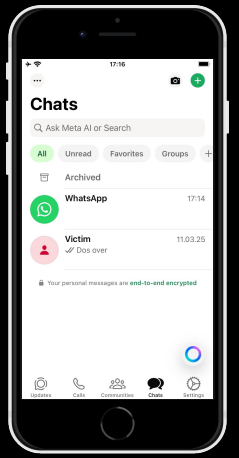
Tracking Device Online Status

- We can track the victim across different companion devices
  - by consistently pinging their devices throughout the day
  - Potentially leaking their daily schedule



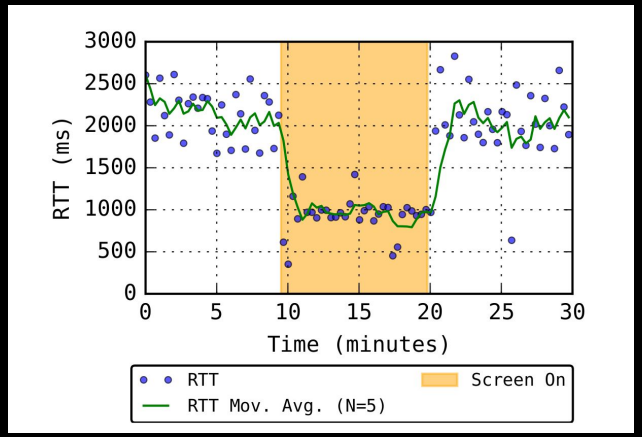
Tracking Device Activity Status

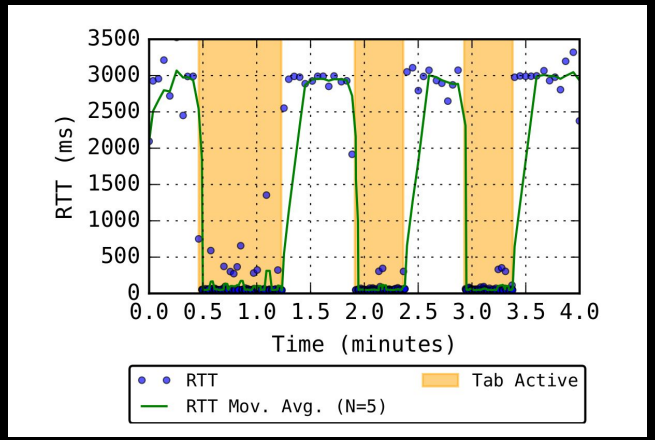
- Hypothesis: different RTT latency in different states
  - Standby (screen off)
  - Active (screen on)
  - Messaging app active (in foreground)



Tracking Device Activity Status

- We can observe the victim's screen/activity state by monitoring relative RTT differences
  - Possible on main and companion devices
    - Pictures: iPhone and WhatsApp Web (Firefox)





DEFCON

Tracking Device Activity Status

- DEMO Time:
  - https://drive.google.com/file/d/1nu8YWU76phX3ZQ9UGHEP1fwG4aFhwxbB

DEFCON

Probing Frequency Limitations

- Signal
  - High-frequency probing triggers rate-limiting
  - One ping every 2 seconds allowed
- WhatsApp
  - No observed limits

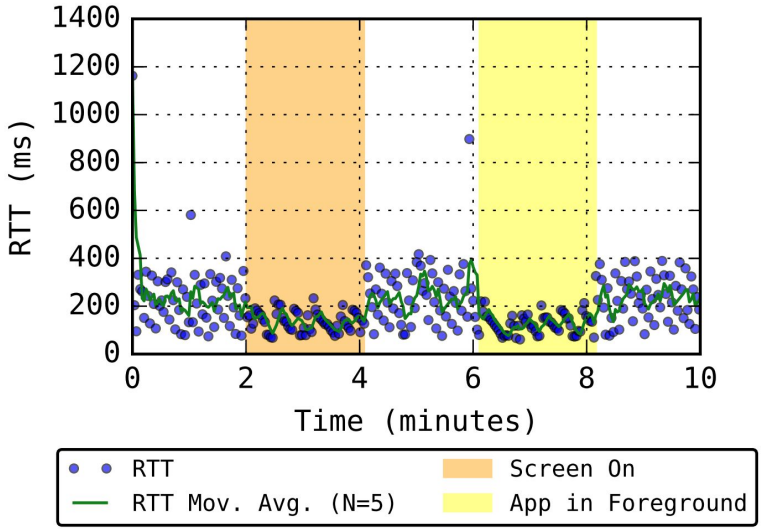
- Different response characteristics for different devices

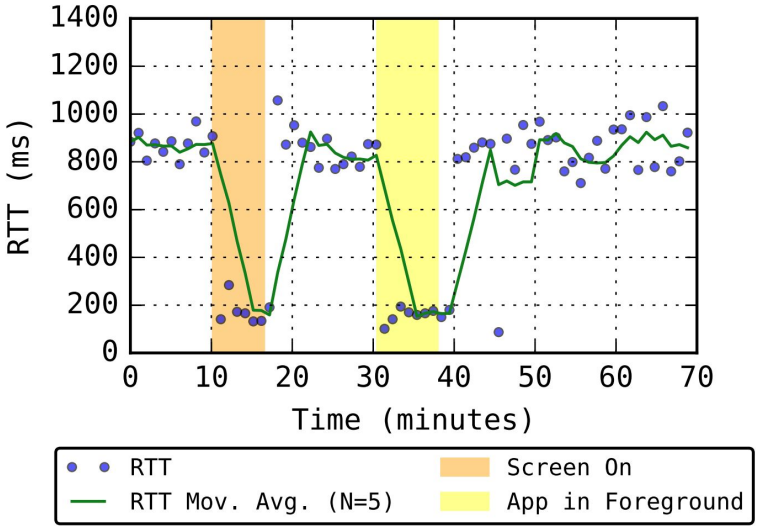
DEFCON

Characteristic Screen ON/OFF Timings

Xiaomi

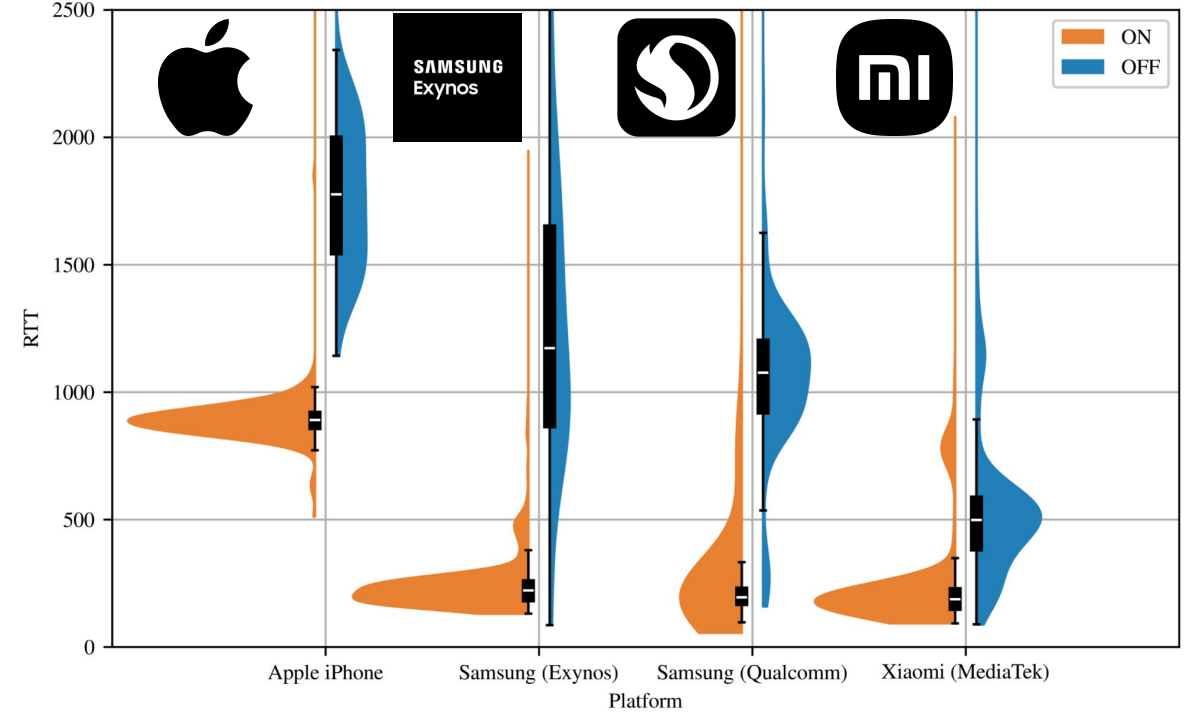
SAMSUNG





DEFCON

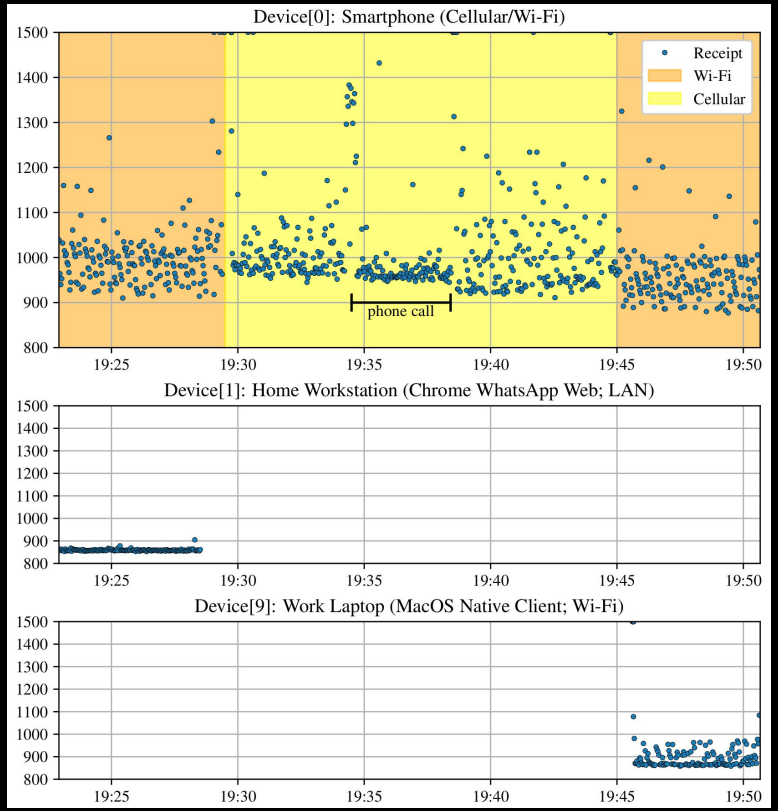
Characteristic Screen ON/OFF Timings



Device models: Apple iPhone 11, Samsung Galaxy A54 5G, Samsung Galaxy S23, Xiaomi Poco M5s

Advanced User Tracking

- Differences in RTTs, based on the current environment (WiFi vs. LTE)
  - Allowing advanced user tracking
  - Certain Phone states caused different patterns e.g. calls



DEFCON

OS Fingerprinting

- Implementation differences can be leveraged to disclose OS of the victim
  - Can be abused for reconnaissance
  - Or weaponized during advanced attacks

	OS	Delivery Receipts	Read Receipts
WhatsApp	Android	Separate	Stacked
	iOS	Separate	Stacked (Reversed)
	Web	Stacked	Stacked
	Windows	Stacked	Stacked
	MacOS	Stacked (Reversed)	Stacked (Reversed)
Signal	Android	Separate	Stacked
	iOS	Separate	Stacked (Random)
	Desktop	Stacked	Stacked (Reversed)

TABLE 5: Deivce/OS Fingerprinting: WhatsApp and Signal show different receipt handling for different platforms.

Resource Exhaustion Attacks

- Sending malformed reactions allows silently exhausting resources at the target device
  - Data Usage: ~ 13 GB per hour
  - Battery Drainage: ~ 18% per hour
  - Attacker is invisible to victim

	Send	Edit	React	Delete	Consumable Data
	65	65	1,000	-	13,320 MB/h
	194	194	194	-	360 MB/h

Privacy Impact

- Attacker knows
  - How many devices
  - Device type (mobile vs. desktop, Android vs. iOS)
  - When they're used
  - How they are used (phone activity, connectivity type)

- Reconnaissance (e.g., to customize exploit)
- Law enforcement (raid suspect when device unlocked)
- Stalking

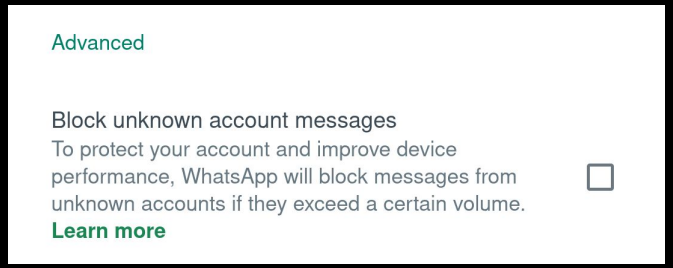
DEFCON

Privacy Impact

- Attacker knows
  - How many devices
  - Device type (mobile vs. desktop,
  - When they're used
  - How they are used (phone activit
- Reconnaissance (e.g., to customize e
- Law enforcement (raid suspect when
- Stalking

Responsible Disclosure WhatsApp

- Reported findings to Meta on September 5th, 2024
  - (Internally) forwarded to relevant engineering team on September 24th, 2024
  - New privacy setting introduced in October 2024



  - Since September 2024, no official/additional information or feedback from Meta :-(

Responsible Disclosure Signal

- Reported findings to Signal on September 5th, 2024
- No response :-(

DEFCON

Issue 2:

Exploiting WhatsApp's Prekey Mechanism

ARTIFACT EVALUATED usenix SECURITY '21 AVAILABLE

ARTIFACT EVALUATED usenix SECURITY '21 FUNCTIONAL

Prekey Pogo: Investigating Security and Privacy Issues in WhatsApp's Handshake Mechanism

Gabriel K. Gegenhuber[1,2], Philipp É. Frenzel[3], Maximilian Günther[1], and Aljosha Judmayer[1]

[1]University of Vienna, Faculty of Computer Science
[2]UniVie Doctoral School Computer Science
[3]SBA Research

Abstract

WhatsApp, the world's largest messaging application, uses a version of the Signal protocol to provide end-to-end encryption (E2EE) with strong security guarantees, including Perfect Forward Secrecy (PFS). To ensure PFS right from the start of a new conversation –even when the recipient is offline– a stash of ephemeral (one-time) prekeys must be stored on a server. While the critical role of these one-time prekeys in achieving PFS has been outlined in the Signal specification, we are the first to demonstrate a targeted depletion attack against them on individual WhatsApp user devices. Our findings not only reveal an attack that can degrade PFS for certain messages, but also expose inherent privacy risks and serious availability implications arising from the refilling and distribution procedure essential for this security mechanism.

1 Introduction

WhatsApp is the world's largest messaging application, with more than 3 billion users worldwide [31]. Under the hood, WhatsApp uses its own version of the Signal protocol for end-to-end encryption (E2EE) of messages [2].

The Signal protocol suite consists of several different protocols [19–22, 27] which together form one of the best end-to-end encrypted communication options available to end users today. Parts of the protocol suite have also been formally analyzed and proven secure in their respective security models [3, 6, 9, 10, 12]. Nonetheless, it remains crucial to continuously analyze protocols in their entirety—including their real-world composition and implementations—to uncover and test new attack strategies, identify real-world limitations, and enhance them accordingly. For our research, we depleted the *ephemeral prekeys* (also *one-time prekeys*) of our test accounts to analyze attacks on *perfect forward secrecy* (PFS) and to highlight novel privacy and availability implications arising from the current replenishing and distribution mechanisms for such prekey bundles.

The importance of one-time prekeys for the PFS of initial messages has already been noted in the specification of Signal's X3DH protocol [22]:

> *"This reduction in initial forward secrecy could also happen if one party maliciously drains another party's one-time prekeys, so the server should attempt to prevent this, e.g. with rate limits on fetching prekey bundles."*

To the best of our knowledge, we are not only the first to test this concrete attack against forward secrecy, but also the first to analyze its feasibility and the general implications of this feature regarding the privacy of users. Hereby, we not only show that WhatsApp currently does not employ any rate limiting on fetching prekey bundles of participants, but also highlight that the lack of a detailed specification on how to handle and replenish ephemeral one-time prekeys, allows for device fingerprinting and gives away the online status of the targeted device. Moreover, extensively querying prekey bundles for a targeted account may cause errors, potentially preventing the retrieval of *any* prekey bundle for that account (even without one-time prekeys). As a result, no one would be able to establish new chat sessions with the victim, leading to an availability issue. While PFS is undoubtedly affected as well, we consider the real world confidentiality impact of this attack to be modest. This is due to the careful design and a clever defense-in-depth strategy of the Signal protocol suite. After being able to circumvent the noise protocol [26], an attacker would still need to get his hands on the long- and medium-term keys of a victim to exploit the lack of forward secrecy and decrypt the previously recorded messages. Even without one-time prekeys, the "self-healing" properties of the double ratchet [27] restore forward secrecy after the first round trip. Nevertheless, the attack on PFS shows that the strong claim from the WhatsApp whitepaper, would at least require a footnote that this currently might not hold for all messages:

> *"Due to the ephemeral nature of the cryptographic keys, even in a situation where the current encryption keys from a user's device are physically compromised, they cannot be used to decrypt previously transmitted messages."* [2]

DEFCON

Background: WhatsApp Prekeys

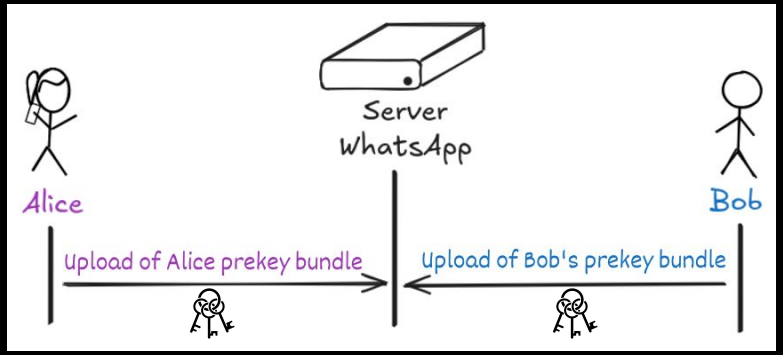
- 3 Layers of Encryption Keys (Key bundle)
  - Identity Key: static, generated during setup
  - Signed Prekey: medium-term, rotated once a month
  - One-Time Prekey: ephemeral, single use

- Multiple keys to ensure Perfect Forward Secrecy (PFS) and Post-Compromise Security (PCS)

- All three keys are combined to initiate a new conversation

Background: WhatsApp Prekeys

- Each client device regularly uploads one-time prekeys to WhatsApp
  - Based on threshold (one-time PK)
  - Or expiration (Signed Prekey)



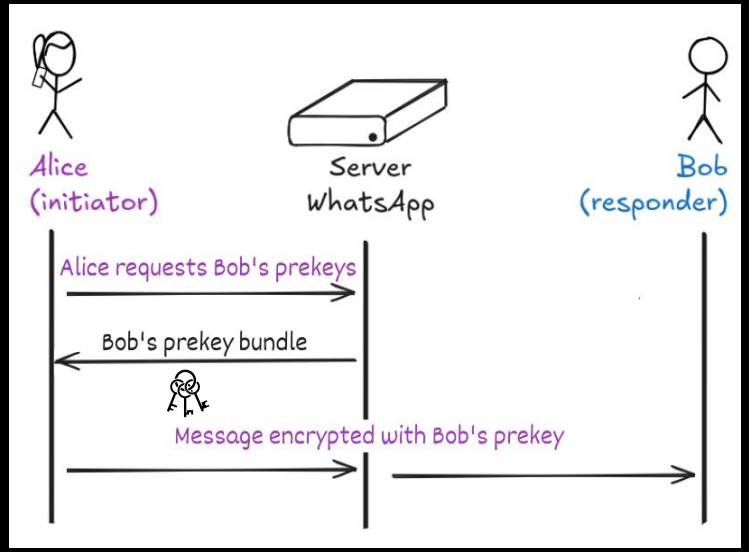
DEFCON

Background: WhatsApp Prekeys

- To start a new conversation:
  1. Alice fetches prekey bundle from server for bob

  2. Alice encrypts the message with bobs prekeys

  3. Bob receives the message and the E2EE channel is established



DEFCON

Question

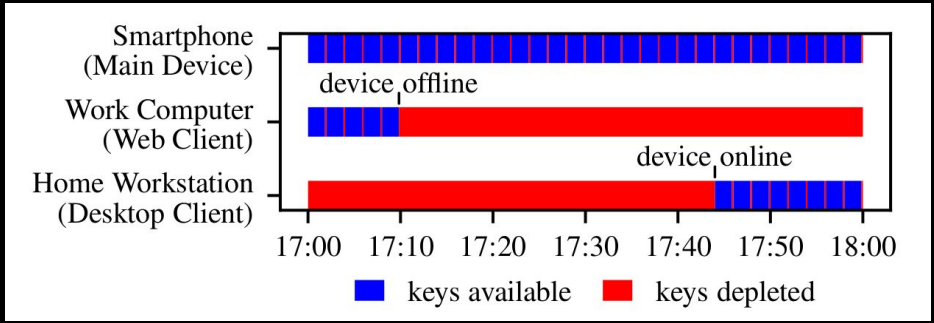
- What happens if there are no one time prekeys?
  - Custom client which repeatedly requests keys
  - Monitor the amount of keys left
  - Detect the moment the keys are refilled

DEFCON

Vulnerabilities

Tracking Device Online Status

- Repeatedly querying prekey bundles drains one-time prekeys on the server
  - Devices can only refill prekeys when they're online
    - Side-channel for a device's online status
  - Attacker solely interacts with WhatsApp servers
    - Cannot be traced back from the victim's side



DEFCON

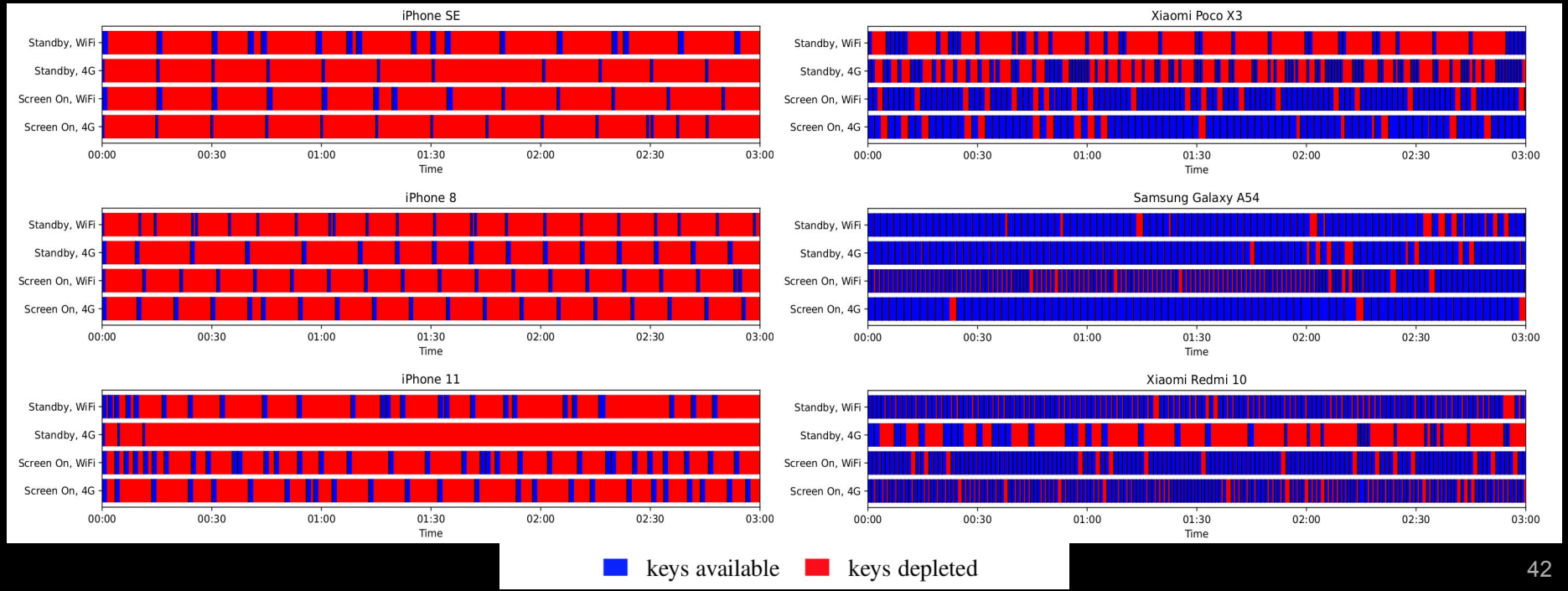
Prekey Depletion Success Rate

- We simulated the attack in practice
  - Different refill behavior based on smartphone model
    - Device fingerprinting
  - Obviously: 100% success rate when device is offline
    - Cannot push fresh prekeys

Device	Standby		Screen On	
	WiFi	4G	WiFi	4G
iPhone SE	85%	94%	90%	93%
iPhone 8	90%	88%	89%	88%
iPhone 11	80%	96%	74%	80%
Poco X3	76%	55%	18%	17%
Galaxy A54	10%	9%	18%	4%
Redmi 10	15%	72%	13%	19%

DEFCON

Characteristic Prekey Refill Behavior



DEFCON

PFS Downgrade

- Attacker can constantly request/deplete a device's one-time prekeys
  - When somebody starts a new conversation with the victim, no one-time prekeys are available on the server
  - Conversation will only be secured by identity key and signed prekey
    - PFS downgrade

DEFCON

OS Disclosure

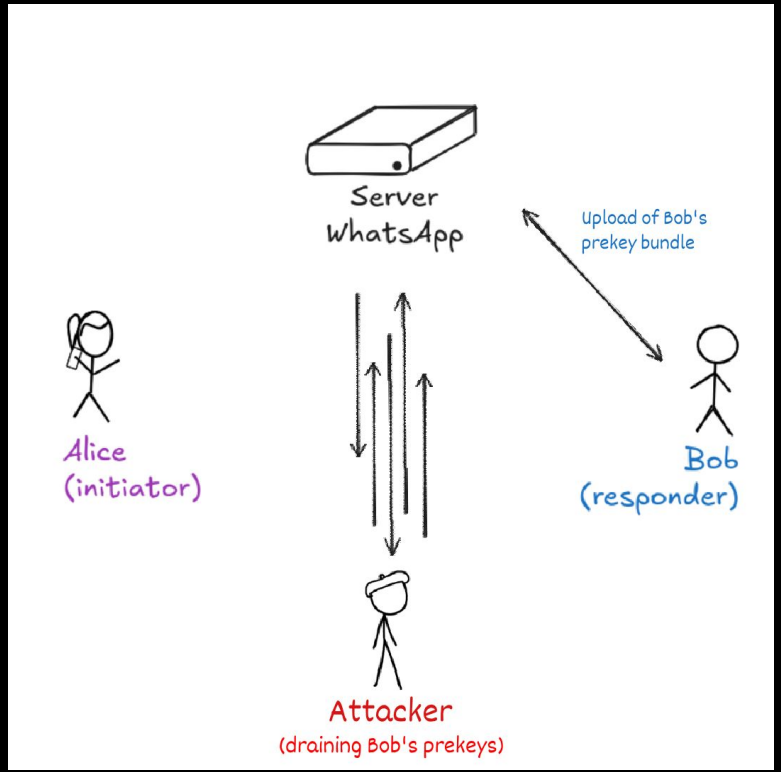
- Different client implementations use different initialization values for their key ID's and batch sizes
  - Can be used to determine the victim's OS

Client Implementation	Initialization Values for *key IDs*			Prekey Batch Size	
	Registration	Signed PK	One-Time PK	Initial	Refill
Android	*R*	0	*R*	812	812
iPhone	*R*	*R*	1	812	812
WhatsApp Web[a]	*R* & 0x3FFF	1	1	200	812
Desktop App macOS	*R*	*R*	1	200	812
Desktop App Windows	*R*	1	1	50	812

R Random number. [a] Verified on Firefox, Chrome, Safari.

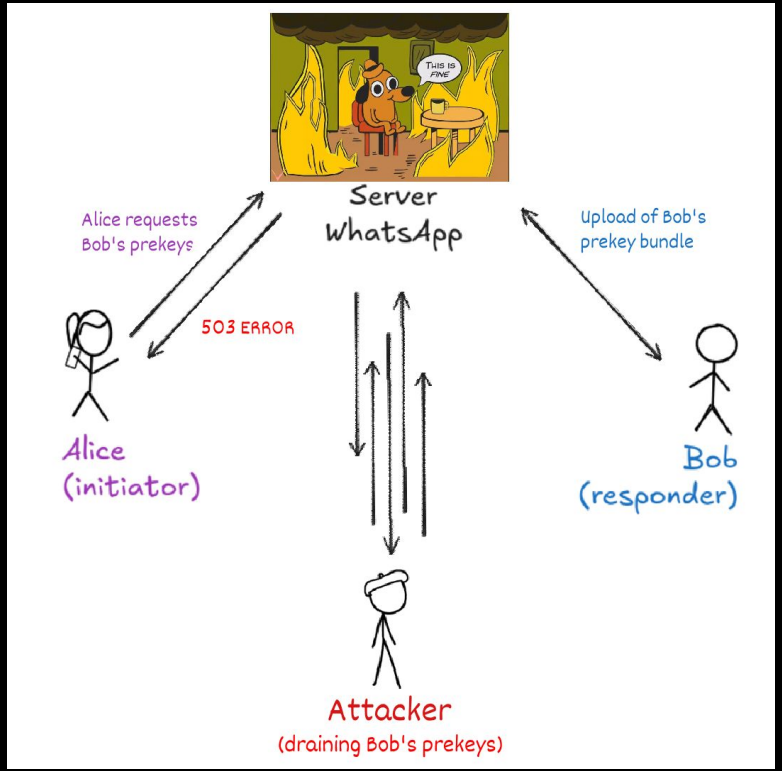
Denial of Service

- Scenario:
  - Attacker draining Bob's one time prekeys
  - Server synchronizes prekey requests



Denial of Service

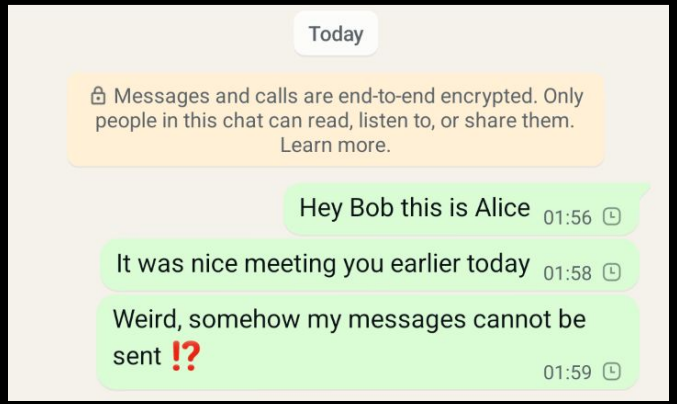
- Scenario:
  - Attacker draining Bob's one time prekeys
  - Server synchronizes prekey requests
- Result:
  - Server runs into concurrency issues
  - Returns 503



DEFCON

Denial of Service

- Issuing many concurrent requests for a specific target device leads to server failures [`503 Service Unavailable`]
  - Hinders anybody from getting the victim's encryption keys
    - Third party is not able to start a new conversation



DEFCON

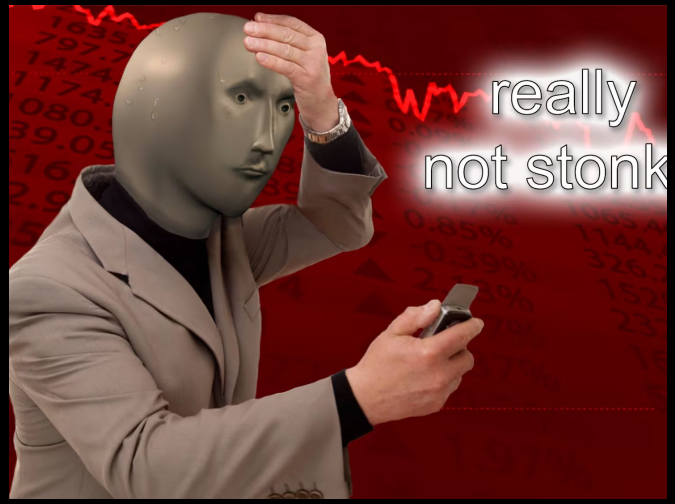
Denial of Service

- DEMO Time:
  - https://drive.google.com/file/d/10Dz8gaMTfvHBnNX3NjZ4qXYfa-53nd2j

DEFCON

Impact DoS

- Hinders arbitrary target from establishing new conversations
  - Inconvenience for User
- For Businesses running WhatsApp
  - No appointments
  - No support
  - No Orders



DEFCON

Impact Privacy Leaks

- Same as previous vulnerabilities
  - Device online state, OS, etc.

- This time Server does the dirty work
  - We don’t need to directly interact with the client

Responsible Disclosure WhatsApp

- Reported findings to Meta on March 28, 2025
- Issue closed on March 30, 2025

DEFCON

Responsible Disclosure WhatsApp

- Reported findings to Meta on March 28, 2025
- Issue closed on March 30, 2025
  - Flagged as duplicate to previous delivery receipt report
  - No answer so far

DEFCON

Mitigations and Key Takeaways

DEFCON

Mitigations

General:
- Rate limiting
- Unifying client behaviour (iOS vs. Android)

Delivery receipt issues:
- Randomize delivery receipt timings
- Improve client-side message validation

Prekey issues:
- Visual indication of missing PFS

DEFCON

Key Takeaways

- Multi-device setups + E2EE amplifies privacy issues
  - Every single device leaks independently
- Due to E2EE, no server-side message validation possible
  - Client-side validation even more important
- Hardened messaging protocols can have security and privacy issues when deployed in practice

Thank You !

DEFCON

Questions ?

Contact

Gabriel

Max

DEFCON

Thank You !

Papers

Careless Whisper

Prekey Pogo