تحلیل و بررسی تماس های VoIP برنامه واتساپ

علی پورقاسمی، اسماعیل پاهنگ، محمد فتاح

مقدمه

در این مقاله ما میخواهیم صدا بر روی پروتکل اینترنت برنامه واتساپ را از جنبه های مختلف بررسی کنیم.

در این تحقیق ما با استفاده از یک دستگاه قفل شکنی(jailbroken) iOS و ابزار هایی که برای تحلیل برنامه های iOS استفاده می‌شوند بکار گرفته شده است که در ادامه این ابزارها را نیز معرفی می‌کنیم.

در این مقاله تماسهای واتساپ را در سه بخش تحلیل می‌کنیم:

۱-تحلیل ترافیک شبکه

۲-تحلیل فایلهای باینری

۳-تحلیل رفتار برنامه در زمان ران تایم

۱ تحلیل ترافیک شبکه

در این بخش مقاله ما میخواهیم به رفتار ترافیک شبکه در طول اجرای یک VoIP در برنامه واتساپ بپردازیم. برای انجام این تحلیل ما از برنامه Wireshark استفاده می کنیم. این برنامه یک برنامه تحلیل کننده متن باز است که برای تجزیه و تحلیل نرم افزارها و توسعه پروتکل های ارتباطی استفاده می‌شود و به وسیله آن می‌توانیم با استفاده از یک رابطه کنترلی تمام ترافیکی که برای آن رابط قابل رویت هستند را مشاهده کنیم.

برای مشاهده کردن رفتارهای ترافیک شبکه در یک دستگاه iOS ما یک رابط شبکه مجازی به وسیله ی وایرشارک می‌سازیم. برای ساختن چنین محیطی ما بر روی دستگاه macOS می‌توانیم از دستور شل زیر استفاده کنیم که در آن در قسمت *<device UUID>* باید شناسه منحصر بفرد جهانی دستگاه iOS خود را وارد کنیم:

```
rvictl -s<device UUID>
```

حال با استفاده از برنامه Wireshark ما رفتار برنامه واتساپ را در طول برقراری یک تماس بررسی می‌کنیم.

Source	Destination	Protocol	Length	Info
2001:4dd6:6c99::7c67:f8e2:ef3c:cd6e	2a03:2880:f23f:c8:face:b00c::7260	TCP	198	63577 → 5222 [PSH, ACK]…
192.168.178.46	10.164.2.83	STUN	72	Binding Request
2001:4dd6:6c99::7c67:f8e2:ef3c:cd6e	2a03:2880:f23f:c8:face:b00c::7260	TCP	121	63577 → 5222 [PSH, ACK]…
157.240.27.52	192.168.178.46	UDP	146	3478 → 63769 Len=118
2a03:2880:f23f:c8:face:b00c::7260	2001:4dd6:6c99::7c67:f8e2:ef3c:cd6e	TCP	121	5222 → 63577 [PSH, ACK]…
2001:4dd6:6c99::7c67:f8e2:ef3c:cd6e	2a03:2880:f23f:c8:face:b00c::7260	TCP	72	63577 → 5222 [ACK] Seq=…
2a03:2880:f23f:c8:face:b00c::7260	2001:4dd6:6c99::7c67:f8e2:ef3c:cd6e	TCP	72	5222 → 63577 [ACK] Seq=…
2a03:2880:f23f:c8:face:b00c::7260	2001:4dd6:6c99::7c67:f8e2:ef3c:cd6e	TCP	72	5222 → 63577 [ACK] Seq=…
fe80::10fb:e2e3:7981:493	fe80::1c93:9035:aeb0:c214	TCP	72	62693 → 63614 [ACK] Seq…
fe80::10fb:e2e3:7981:493	fe80::1c93:9035:aeb0:c214	TCP	336	62693 → 63614 [PSH, ACK…
fe80::1c93:9035:aeb0:c214	fe80::10fb:e2e3:7981:493	TCP	72	63614 → 62693 [ACK] Seq…
192.168.178.46	157.240.27.52	UDP	137	63769 → 3478 Len=109
192.168.178.46	80.187.102.55	STUN	72	Binding Request
157.240.27.52	192.168.178.46	UDP	150	3478 → 63769 Len=122
157.240.27.52	192.168.178.46	UDP	150	3478 → 63769 Len=122
192.168.178.46	157.240.27.52	UDP	139	63769 → 3478 Len=111
157.240.27.52	192.168.178.46	UDP	134	3478 → 63769 Len=106
192.168.178.46	157.240.27.52	UDP	134	63769 → 3478 Len=106
157.240.27.52	192.168.178.46	UDP	140	3478 → 63769 Len=112
192.168.178.46	157.240.27.52	UDP	137	63769 → 3478 Len=109
192.168.178.46	10.164.2.83	STUN	72	Binding Request
157.240.27.52	192.168.178.46	UDP	133	3478 → 63769 Len=105
192.168.178.46	80.187.102.55	STUN	72	Binding Request
192.168.178.46	157.240.27.52	UDP	141	63769 → 3478 Len=113
157.240.27.52	192.168.178.46	UDP	142	3478 → 63769 Len=114
192.168.178.46	157.240.27.52	UDP	150	63769 → 3478 Len=122
192.168.178.46	157.240.27.52	UDP	130	63769 → 3478 Len=102

همانطور که مشاهده می کنید در Wireshark تمامی بسته های شبکه ایی که در طول این تماس به شبکه فرستاده شدند و یا از آن برگردانده شده اند را به ما نشان می‌دهد و این بسته ها (packet ها) عموما از سه نوع پروتکل تشکیل شده اند. یکی از پروتکل های استفاده شده در طول تماس پروتکل (Session Traversal Utilities) STUN است.

همانطور که میدانید تماسهای برقرار شده در برنامه واتساپ به صورت همتا به همتا (peer-to-peer) انجام می‌شوند و دو همتا در یک ارتباط همتا به همتا به آی پی های یکدیگر نیاز دارند ولی مشکلی که در اینجا بوجود میاید اینست که اکثر کاربران واتساپ در پشت شبکه های نت NAT قرار دارند.

امروزه استفاده از نت NAT بسیار فراگیر شده به گونه ایی که اکثر ما که در فضای اینترنت جستجو می‌کنیم اغلب پشت چنین لایه NAT قرار داریم. NAT روشی برای تغییر آدرس و پورت در هدر لایه آی پی IP است به این صورت که یک دستگاه که نت NAT روی آن پیاده سازی شده است آدرس بسته هایی را که از یک درگاه

دریافت می‌کند را تغییر می‌دهد و یک آدرس جدید روی آن می‌گذارد. این آدرس هم می‌تواند آدرس فرستنده بسته باشد و یا آدرس گیرنده آن.

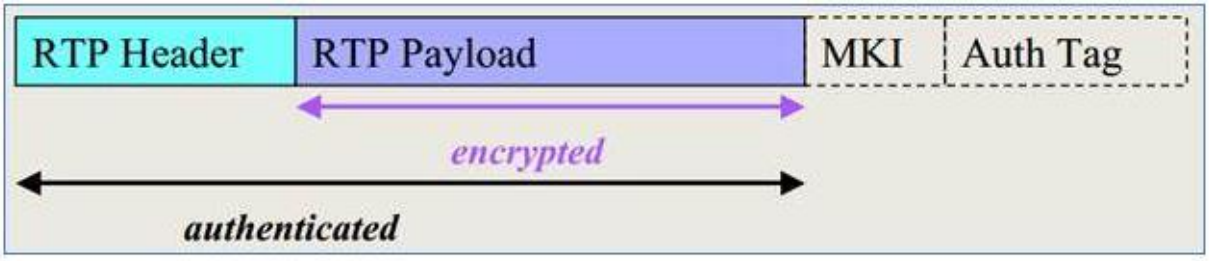
حال مشکلی که برای ارتباطات همتا به همتای مانند تماس در شبکه واتساپ بوجود می آید وقتی که همتا ها در پشت شبکه NAT قرار دارند این است که ممکن است همتاها آدرس آی پی IP ثابتی نداشته باشند. از سویی حتی اگر آدرسی که NAT به یک همتا داده باشد را داشته باشیم در خیلی از مورد چون همتای دیگری نشستی با ما آغاز نکرده است نمی‌توانیم با او ارتباط برقرار کنیم. پس نیاز به مکانیزم هایی داریم که این محدودیت ها را دور بزنند.

بسته ها (Pockets) با پروتکل STUN قدم های لازم را برای برقراری یک ارتباط همتا به همتا بین دو کلاینت(Client) پشت نت NAT انجام می‌دهد.

بسته های شبکه ی دیگری که در این ارتباط استفاده می‌شوند بسته های TCP هستند. که این بسته ها از سمت کلاینت واتساپ به سرورهای واتساپ فرستاده می‌شود و برعکس. و بسته های UDP بین دو همتای این ارتباط که شامل تماس گیرنده و فردی که با او تماس گرفته می‌شود هستند تبادل می‌شوند و در واقع این بسته های UDP همان بسته هایی هستند که اطلاعات موجود در یک تماس را در اختیار دارند. همانطور که می‌دانید پروتکل های UDP برخلاف TCP سرویس غیر مطمئنی ارائه می‌دهند که ممکن است بسته داده ها در طول ارتباط از بین بروند و به مقصد نرسند.

در وایت پیپر واتساپ اشاره شده است که برای ارتباط از بسته Secure Real Time Protocol (SRTP) استفاده می‌کنند و این بسته های SRTP که شامل اطلاعات یک تماس هستند همان بسته های UDP اند.

پروتکل SRTP مشخصه ایی از پروتکل RTP تعریف می‌کند که هدف آن ارائه رمز گذاری، احراز هویت پیام، یکپارچگی و حفاظت از داده های موجود در آن بسته است. در حالت کلی بسته های RTP به صورت زیر فرستاده می‌شوند.

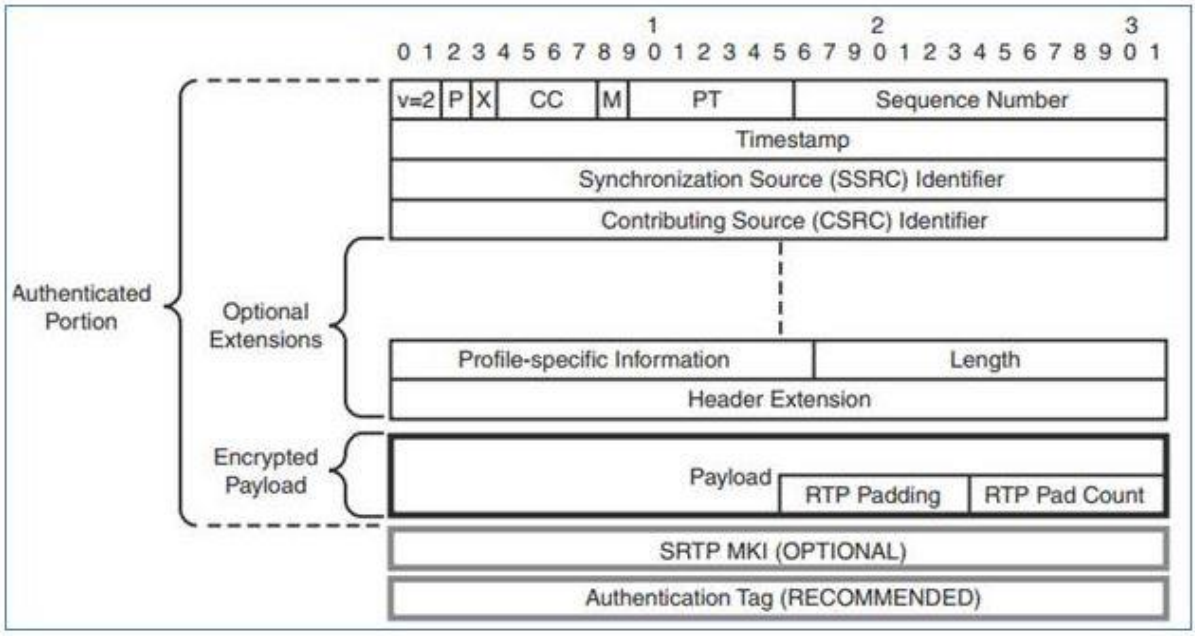


لینک یک پیام SRTP هم باید در همان فرمت پیام RTP باشد ولی در هدر آن دو سمت اضافه وجود دارد.

۱- Master key identifier (MKI)

۲- Authentication tag

فرمت کلی آن ها به صورت زیر است.



در شکل زیر یک بسته SRTP را مشاهده می‌کنیم که بصورت لیستی از اعداد دسیمال Decimal نمایش داده شده اند و توسط تماس گیرنده به سمت فردی که با تماس حاصل شده فرستاده می‌شود.

```
80 78 00 1e 00 02 8f 00 f5 d2 f5 cf 69 a3 8c 54 22 d3 69 8d 05 db 65
de cb 38 d4 8b 09 bc a9 9a 64 92 61 51 c2 49 3b 79 52 3b 11 52 10 ba
56 c0 2a ba 44 9e 5d b6 bb e4 cd d0 f4 72 d8 13 48 31 0e 4c b9 d8 7d
e2 09 72 f1 c9 26 97 15 bb 6b 00 9d 7e fb ec 3f a2 70 01 9b f1 f2 83
71 91 a9 60 fc 5f 34 3a b6 89 8c b7 3e 42 b2 79 ce 37 fd 7c ef 65 94
ed 1d 56 f9 2e 4c 35 e9 fe b9 bd 3d 87 1e 95
```

با توجه به فرمت SRTP 4بایت اول بسته ، هدر آن است که در این شکل هدر این بسته با رنگ قرمز نشان داده شده است و ما این بسته را بصورت باینری مینویسم تا اعداد آن را استخراج کنیم.

```
0x8078001e =
0b10_0_0_0000_0_111100_00000000000011110 =
V=10|P=0|X=0|CC=0000|M=0|PT=111100|SEQ=00000000000011110
```

همانطور در نمایش باینری هدر این بسته مشاهده می کنید دو عدد اول این بسته نشان دهنده ورژن این بسته هستند که در این بسته این ورژن برابر دو ۲ است. بیت سوم پدینگ را نشان می‌دهد از آنجا که P=0 به این معنی است که این بسته هیچ پدینگی padding ندارد. چهارمین بیت آن نشان دهنده extension bit است.

از آنجا که این عدد برابر صفر است یعنی هیچ بخش دیگری به هدر این بسته RTP اضافه نشده ست طول آن برابر همان طول بسته RTP است. چهار بیت بعدی نشان دهنده تعداد CSR هستند. بیت بعدی بیت M یا همان maker است که برابر صفر است، اگر این بیت برابر ۱ باشد نشان دهنده آن است که اطلاعات موجود در این بسته یک ارتباط ویژه ایی با خود برنامه دارد. (خود برنامه واتساپ) شش بیت بعدی نشان دهنده نوع بسته هستند که در این مثال برابر ۶۰ است. (PT=Pay load type) منظور از نوع بسته RTP یا SRTP بودن بسته نیست این مقدار توسط خود برنامه استفاده می‌شود، یعنی اینکه خود برنامه واتساپ ممکن است برای ارتباط های تماس خود چندین نوع بسته مختلف استفاده کند که توسط خود برنامه تعیین شده است. و ۱۷ بیت پایانی نشان دهنده ترتیب بسته ها هستند. بسته های اینترنت ممکن است در طول ارتباط جابجا شوند و به ترتیب به مقصد نرسند. به همین دلیل مقدار SEQ به هر بسته یک عددی اختصاص می دهد که این عدد نشان دهنده توالی این بسته هاست و وقتی که این بسته ها به مقصد برسند کلاینت دیگر این بسته ها را به ترتیب منظم می‌کند و به ترتیب آنها را می‌خواند.

نکته قابل توجه در قسمت SEQ بسته های واتساپ این است که SEQ ها برخلاف اکثر بسته های اینترنت از صفر شروع می‌شوند. (همانطور که می‌دانید اکثر بسته های اینترنتی برای شروع یک عدد رندوم (تصادفی) را به SEQ اولیه می‌دهند.)

۴بایت بعدی موجود در این بسته نشان دهنده برچسب زمانی (time stamp) این بسته هستند.

در VoIPها، RTP برچسب های زمانی برای بسته های صوتی تعیین می‌کند تا بتوان آنها را توسط گیرنده بافر کرده دوباره سرهم کرد و بدون خطا تحویل داد.

چهار بایت پس از آن که با رنگ سبز نشان داده شده است. نشان دهنده SSRC هستند. که این عدد برای ایجاد تمایز بین تماس هایی که بصورت همزمان در حال انجام هستند استفاده می‌شوند.

حال که ساختار یک بسته UDP را مشاهده کردیم و از آنجا که این بسته های UDP ساختار SRTP دارد مطمئن می‌شوم که واتساپ از پروتکل SRTP برای تماس های خودش استفاده می‌کند و همینطور با مشاهده آی پی IP بسته TCP مطمئن می‌شویم که بسته های TCP برای ارتباط با سرور های واتساپ استفاده می‌شوند. این بسته ها بصورت Noise Pipes Protocol رمزگذاری می‌شوند که در ادامه آنها را بررسی می‌کنیم.

۲ تحلیل فایل های باینری

فایل های برنامه واتساپ در iOS از دو بخش اصلی تشکیل شده اند:

۱-فایل های برنامه واتساپ

۲- چارچوپ هسته واتساپ

در این قسمت ما این فایل های باینری را از طریق برنامه radar2 بررسی می‌کنیم. همچنین از آنجا که تمام فایل های دانلود شده از اپ استور، رمزگذاری می‌شوند ما برای خواندن فایل های باینری واتساپ نیاز به رمزگشایی این فایل ها داریم. برای رمزگشایی این فایل ها ما از برنامه bfdecrypt استفاده می‌کنیم.

پس از مشاهده این فایل های باینری توسط bfdecrypt، متوجه می‌شویم که واتساپ از چه الگوریتمی در فایل های باینری خود استفاده می‌کند که در ادامه هرکدام از آنها را توضیح می‌دهیم و تحلیل می‌کنیم.

۲.۱ libSignal protocol c

واتساپ از کتابخانه متن باز libSignal protocol c استفاده می‌کند، که این کتابخانه برای پیاده سازی پروتکل سیگنال است. پروتکل سیگنال یک پروتکل رمزنگاری غیر فدارسیونی است که می‌تواند برای ارائه رمزگذاری سرتاسر (end-to-end encryption) برای تماس های صوتی، تماس های ویدیویی و پیام های فوری استفاده شود.

این پروتکل بر پایه الگوریتم double ratchet است که رمزگذاری فایل های واتساپ را انجام می‌دهد. قسمت زیر یک فایل باینری موجود در پیام های واتساپ است که اگر به آن دقت کنید متوجه می‌شوید که تابع های استفاده شده در این فایل باینری همان تابع های موجود در کتابخانه libSignal Protocol c- هستند.

```
r2 WhatsAppCore
[0x0082b517]> / _Signal_
Searching 8 bytes in [0x0-0x654000]
hits: 33
0x00837a7b hit2_0 .il_key_data_from_Signal_keydispatch_.
0x0083df33 hit2_1 ._torlice_Signal_protocol_paramet.
0x008407c0 hit2_2 .d_fac_3key_Signal_message_big.
0x00840d50 hit2_3 .mmetric_Signal_protocol_paramet.
0x00840e70 hit2_4 .ob_Signal_protocol_paramet.
0x00841492 hit2_5 .pre_key_Signal_messagesigna.
0x008de24b hit2_6 .agc_reset_alice_Signal_protocol_paramet.
0x008de274 hit2_7 .rs_create_alice_Signal_protocol_paramet.
0x008de440 hit2_8 .bitno_MRDTX_bob_Signal_protocol_paramet.
0x008de467 hit2_9 .ters_create_bob_Signal_protocol_paramet.
0x008e311c hit2_10 .pre_big_pre_key_Signal_message_copy_pr.
0x008e3139 hit2_11 .ge_copy_pre_key_Signal_message_create_.
0x008e3158 hit2_12 ._create_pre_key_Signal_message_deserial.
0x008e317c hit2_13 .rialize_pre_key_Signal_message_destroy.libsrtp
...
```

libsrtp ۳ . ۲

با ادامه خواندن فایل های واتساپ توسط ابزار bfdecrypt ما نوع جدیدی از فایل ها را مشاهده می‌کنیم که با دقت در نام کتابخانه های استفاده شده در این فایل متوجه می‌شویم که واتساپ از کتابخانه libsrtp استفاده می کند. و این کتابخانه برای پیاده سازی بسته های SRTP واتساپ هستند. دلیل این ادعا این است که اگر به فایل زیر در واتساپ نگاه کنیم.

```
r2 WhatsApp
0x1001ada34 / <[libsrtp
0x100ee5546 hit1_0 .rc %08XUnknown libsrtp error %duns.
0x100ee57eb hit1_1 .d to initialize libsrtp!. :sFailed to r.
0x100ee580a hit1_2 .led to register libsrtp deinit.Failed.
```

0x100ee5831 hit1_3 .to deinitialize **libsrtp**٪ .sAES_CM_128._
0x100ee5883 hit1_4 .ck crypto Init **libsrtp** .create pool. .
0x100f07b80 hit1_5 . packet: %s**libsrtp**stat test%s: c.

متوجه می‌شویم که در این فایل ها از رشته هایی استفاده شده است که اگر به کد خود libsrtp در گیتاب نگاه کنیم، این رشته ها در آنجا هم تکرار شده اند. مانند رشته ی "cloning stream (SSRC: 0x%08x)" در کد زیر که در گیت هاب موجود است.

pjsip ۴ . ۲

طبق گزارشاتی که بالاتر از برنامه Wireshark بدست آوردیم، برنامه واتساپ از بسته های STUN برای کنترل NAT استفاده می‌کند. واتساپ برای پیاده سازی این بسنه های STUN از کتابخانه pjsip استفاده ی کند.

دلیل این حرف ما این است که اگر بعضی از فایل های موجود در واتساپ را نگاه کنیم مانند فایل زیر در آن رشته هایی را مشاهده می‌کنیم که در کد موجود در کتابخانه pjsip در گیتاب قرار دارند.

r2 WhatsApp
]0x1013ddb4f / <[**pjmedia**
Searching 7 bytes in [0x100000000-0x100fb4000]
hits: 180
0x100edd55f hit9_0 .io_piggyback.cc**pjmedia**_audio_piggyback.
0x100edd591 hit9_1 .r %d, stream %p**pjmedia**_audio_piggyback.
0x10٠edd5d4 hit9_2 .d, tx_packet %d**pjmedia**_audio_piggyback.
0x100edd601 hit9_3 .ideo_enabled %d**pjmedia**_audio_piggyback.
0x100eddcf3 hit9_4 .ibyuv converter**pjmedia**_converter_creat.
0x100eddd21 hit9_5 .rter count = %d**pjmedia**_converter_creat.
0x100ede3e3 hit9_6 .rame, status=%d**pjmedia**_delay_buf_get_s.
0x100ede46e hit9_7 .%sec_delay_buf**pjmedia**_echo_create2٪ .
0x100ede64d hit9_8 .eUnknown **pjmedia**-videodev error.
0x100ede90c hit9_9 .o errorUnknown **pjmedia**-audiodev error.
0x100edebba hit9_10 .ATENCY)Unknown **pjmedia** error %dUnspec.
0x100ee027e hit9_11 .queue.format.c**pjmedia**_format_get_vide.

0x100ee02ca hit9_12 .mat info for %d**pjmedia**_format_get_vide.
0x100ee1446 hit9_13 .c_buf too short**pjmedia**_h26x_packetize.
...

۳ تحلیل برنامه در زمان اجرا

در این قسمت از تحقیق خود ما میخواهیم رفتار برنامه واتساپ را در زمان اجرا بررسی کنیم، برای اینکار ما از ابزاری به اسم Frida استفاده می کنیم.

این برنامه چندین هوک جاوا اسکریپ برای برنامه های موبایل میسازد تا رفتار توابع موجود در یک برنامه را بررسی کند و پارامترهایی که توسط یک تابع فراخوانده (بارگردانده) می‌شوند را مشاهده و بررسی کند تا بتوانیم رفتار توابع موجود در یک برنامه را تحلیل کنیم.

۳ . ۱ کلید انتقال

طبق وایت پیپر منتشر شده توسط واتساپ رمزنگاری تماس های واتساپ توسط یک کلید اصلی (Master key)۳۲بیتی SRTP انجام می‌شود که این کلید بصورت رندوم توسط تماس گیرنده ساخته می‌شود. این کلید اصلی رمزنگاری می‌شود و به سمت فرد دیگری که تماس گیرنده قصد تماس با آن را دارد (مخاطب تماس) فرستاده می‌شود. حال ما در این قسمت می‌خواهیم با استفاده از برنامه Frida توابعی که قصد انجام این کار را دارند را بیابیم. برای اینکار ما توابعی که در آنها کلمه Secret بکار رفته شده است را دنبال می‌کنیم و از کد زیر استفاده می‌کنیم:

```
Frida-trace -U WhatsApp -m"[*secret* *]*" m- "[*Secret* *]*"
```

پس از انجام یک تماس در برنامه Frida نوشته زیر را به ما تحویل می‌دهد.

```
]+WAHKDF
deriveSecretsFromInputKeyMaterial :0x121e08a20
salt0 :x0
info :0x121e07840
outputLength :0x2e
withMessageVersion :0x3
[
```

در اینجا چیزی که Frida به ما میگوید این است که متد deriveSecretsFromInputKeyMaterial کلاس WAHKDF فراخوانده شده است و ورودی هایی که به آن داده شده است 0x121e08a20 و 0x121e07840 اند. این اعداد درواقع پوینتر اشاره گر به ورودی ها هستند.

برنامه Frida به ما اجازه می‌دهد که یک objective C را از یک pointer (نشانه گر) مشخص در جاوا اسکریپ بسازیم.

برای بررسی بیشتر تابع deriveSecretsFromInputKeyMaterial ما از تابع هوک زیر استفاده می‌کنیم تا بتوانیم تحلیل بیشتری برای ورودی ها و خروجی های این تابع داشته باشیم.

```
{
  onEnter: function (log, args, state) {
  log("+[ WAHKDF deriveSecretsFromInputKeyMaterial: " +
     ObjC.Object( args[2] ).toString() + "\n" +
   " salt: " + ObjC.Object( args[3] ).toString() + "\n" +
   " info: " + ObjC.Object( args[4] ).toString() + "\n" +
   " bytes : " +    args[5].toInt32 () + "\n" +
   " withMessageVersion : " + args[6].toInt32 () + "\n]");
  }
 }
```

خروجی این تابع بصورت زیر است:Fridaپس از برقراری تماس و فراخوانی این تابع توسط

```
]+WAHKDF deriveSecretsFromInputKeyMaterial۰۹> :a38e76 fe90e4f1 26ed66d0
5a6783ba d48776b6 1daaf7c9 39c005ea 2d8ccdf6 <
salt : nil
info۳۰۴۰۷۳۲ ۳۷۳۱۳۶۳۲ ۳۹۳۰۳۵۳۷ ۳۴۳۹۳۱۳۵> : e 77686174 73617070 2e6e6574 <
bytes ۴۶ :
withMessageVersion۳ :
[
```

اگر به خروجی ها دقت کنیم، مشاهده می‌کنیم پارامتر های ورودی و خروجی این تابع یک شیء NSData هستند. NSData یک کلاس موجود در سویفت هستند که شامل بافری از بایت های استاتیک هستند

اگر دقت کنید متوجه می‌شویم که طول پارامتر اول ۳۲ بایت است که دقیقا خصوصیات کلید اصلی توضیح داده شده در واتساپ را دارند که ما به این نتیجه میرسیم که پارامتر اول این تابع همان کلید اصلی تماس است و پارامتر سوم این تابع یک رشته ASCII است که نشان دهنده jabber identifiers تماس گیرنده است می-کنیم.

۳ . ۲ رمزگذاری کلید اصلی

همانطور که در وایت پیپر واتساپ آمده است در طول تماس ما از کلید اصلی (Master key) برای فرستادن پیام های تماس استفاده می‌کنیم و همانطور که در قسمت بالا مشاهده کردیم پیام ها را توسط این کلید اصلی تغییر می‌دهیم تا اگر فردی در این بین پیام ها را شنود کند نتواند این پیام ها را بخواند. واضح است با داشتن کلید اصلی توسط فرد دیگری و شنود کردن پیام های فرستاده شده بین دو طرف تماس می‌تواند به راحتی پیام های ارسال شده توسط این دو نفر را شنود کند و به آنها دسترسی داشته باشد. پس کلید صلی به این راحتی نباید به دست فرد دیگری باشد و وقتی که یک تماس برقرار می‌شود فرستادن کلید اصلی از تماس گیرنده به فرد دیگر باعث پایین آمدن امنیت تماس می‌شود. به همین دلیل ما تصور می‌کنیم که واتساپ برای فرستادن کلید اصلی به سمت فرد دیگر از تماس گیرنده این کلید اصلی را رمز گذاری می‌کند و به سمت فرد دیگر می‌فرستد. به همین دلیل ما ار ابزار Frida ما از توابع که در آنها لفظ crypt استفاده شده است استفاده می‌کنیم.

پس کد زیر را وارد Frida می‌کنیم:

```
Frida-trace -U WhatsApp -m"*crypt*" i- "[*crypt* *]*"
```

وقتی که تماس برقرار می‌شود تابع Signal_encrypt از کتابخانه libSignal-protocol-c صدا زده می‌شود از آنجا که این کتابخانه متن باز است (بالاتر راجع به این کتابخانه صحبت کرده ایم) می‌توانیم به کدی که در این کتابخانه موجود است دسترسی داشته باشیم.

ورودی های این کتابخانه بصورت زیر است و خروجی آن یک عدد صحیح (integer) است.

```
int Signal_encrypt(Signal_context
*context,
                                    Signal_buffer **output,
                                    int cipher,
                                    const uint8_t *key, size_t key_len,
```

```
const uint8_t *iv, size_t iv_len,
 const uint8_t *plaintext, size_t
plaintext_len);
```

قسمت متن آشکار پارامتر توسط Frida که تابع Signal_encrypt را هوک می‌کند خوانده می‌شود. متن آشکار یا متن رمزنگاری نشده هرگونه اطلاعات قابل فهم برای دو طرفی هست که فرستنده قصد انتقال آن را به گیرنده دارد.

عکس زیر نشان دهنده قسمت رمزنگاری نشده است که از Frida بدست میاوریم .

```
52 22 0a 20 09 a3 8e 76 fe 90 e4 f1 26 ed 66 d0 5a 67 83 ba d4 87 76
b6 1d aa f7 c9 39 c0 05 ea 2d 8c cd f6 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d
0d 0d 0d
```

و خروجی تابع Signal_encrypt بصورت زیر است.

```
fb 69 1b 80 c1 e6 12 09 48 f1 9f 63 fc 2c 05 96 d9 71 6e 62 2f 5d 41
d4 0b c6 44 b8 16 79 df 66 2c ec fc 0d 64 50 3f d0 02 2f 67 f1 2d 33
ef 64 70 dd 8f 13 1b 13 87 56 82 47 80 a8 1f b5 4d 80
```

اگر به ساختار متن آشکار این تابع نگاه کنیم، متوجه می‌شویم که ۴بایت اول برای سریالیزه کردن (سریالی کردن) کلید اصلی توسط پروتکل بافر استفاده می‌شود. پروتکل بافر یک روش سریالیزه کردن اطلاعات است.

پس از آن بایت هایی که با رنگ آبی مشخص شده همان کلید اصلی هستند و ۱۳ بایت آخر نشان دهنده پدینگ padding رمزنگاری هستند.

با خواندن فایل های libSignal-protocol-c متوجه می‌شویم که این رمز نگاری توسط الگوریتم Double Ratchet استفاده می‌شود که بخشی از پروتکل سیگنال است.

همانطور که بالاتر هم نیز اشاره شده بود ما به یک بسته SRTP علاوه بر قسمت RTP قسمت MKI (Master key identifier) تگ را نیز اضافه می کنیم. به همین دلیل است اگر به خروجی دقت کنیم متوجه می‌شویم که بایت های بیشتری نسبت به ورودی دارد زیرا این بخش ها به پیام در همین تابع اضافه می‌شوند.

۳ . ۳ آماده سازی کلید اصلی

حال میخواهیم ببینیم که برنامه واتساپ با این کلید اصلی رمزگذاری شده چه عملیاتی را انجام می‌دهد. به همین دلیل به دنبال توابعی هستیم که در آنها کلمه Signal بکار رفته شده باشد و آنها را پیدا می‌کنیم.

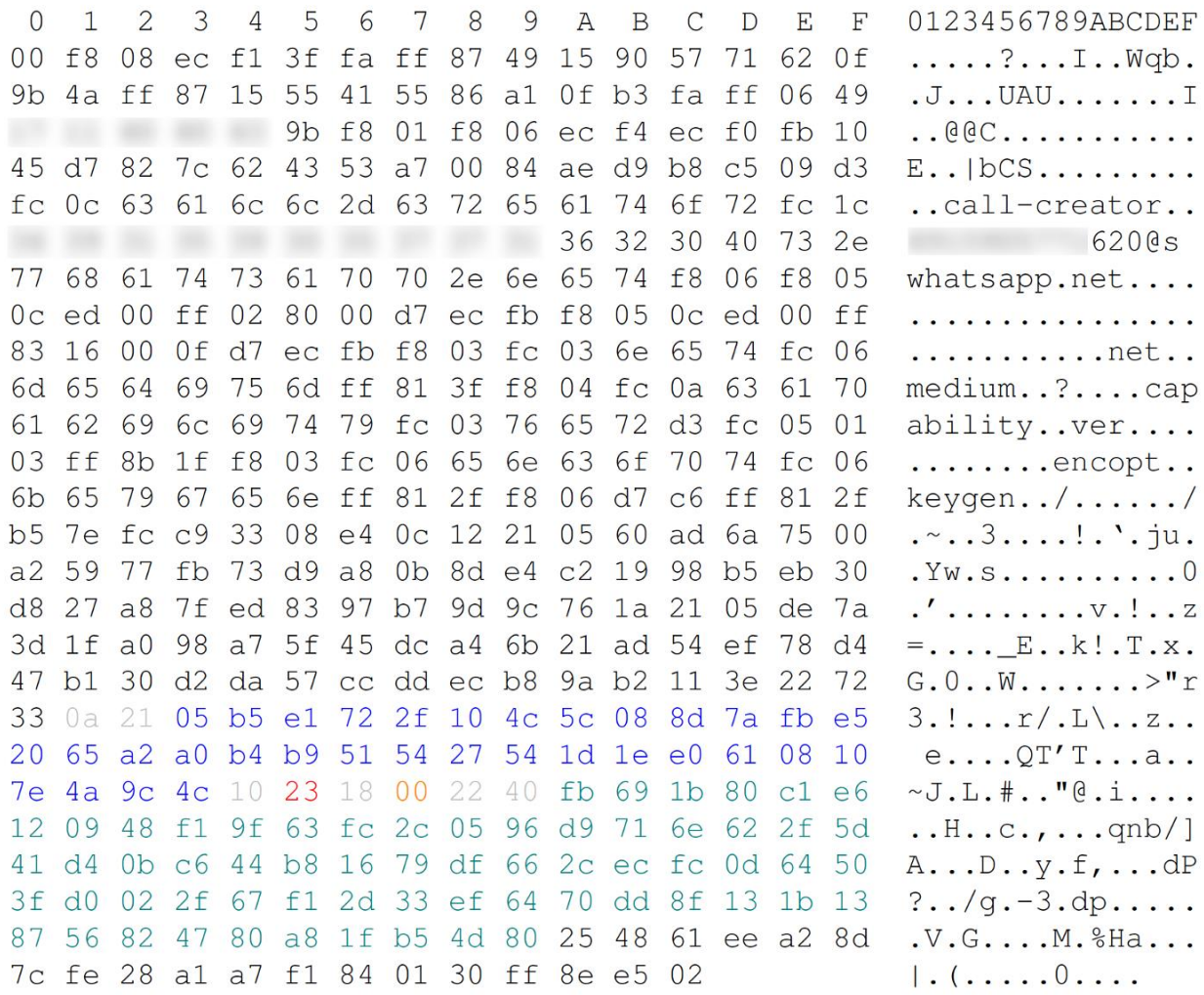
```
Frida-trace -U WhatsApp -i"*Signal*"
```

و با وارد کردن دستور بالا Frida به ما تابع textsecure__Signal_message__pack را معرفی می کند. این تابع شامل پارامترهای مورد نیاز برای رمزگذاری و رمزگشایی کلید اصلی که توسط پروتکل سیگنال تولید می‌شود را در بردارد.

```
0a 21 05 b5 e1 72 2f 10 4c 5c 08 8d 7a fb e5 20 65 a2 a0 b4 b9 51 54
27 54 1d 1e e0 61 08 10 7e 4a 9c 4c 10 23 18 00 22 40 fb 69 1b 80 c1
e6 12 09 48 f1 9f 63 fc 2c 05 96 d9 71 6e 62 2f 5d 41 d4 0b c6 44
b8 16 79 df 66 2c ec fc 0d 64 50 3f d0 02 2f 67 f1 2d 33 ef 64 70
dd 8f 13 1b 13 87 56 82 47 80 a8 1f b5 4d 80
```

عکس بالا خروجی textsecure__Signal_message__pack است. حال ما می‌خواهیم این خروجی را تحلیل کنیم. دو بایت اول این خروجی که با رنگ خاکستری مشخص شده اند مربوط به سریالیزه کردن پیام سیگنال هستند. کلید Ratchet فرستنده هستند. بایت قرمز نشان دهنده شمارنده پیام قبلی است. بایت های سبز نشان دهنده کلید اصلی رمز گذاری شده هستند که همانطور که مشاهده می‌کنید دقیقا برابر مقدار خروجی تابع Signal encrypt که پیشتر راجع به آن صحبت شده است.

وقتی که توابع objective c مرتبط با XMPP را دنبال کنیم، متوجه می‌شویم که تابعی بنام writeNoiseFrameToSocketWithPayload از کلاس XMPPStream صدا زده می‌شود. این تابع یک پیام XMPP را بصورت رمزگذاری شده توسط پروتکل noise pipe در بسته های TCP به سرورهای واتساپ پیام هایی را می‌فرستد. حال قصد مشاهده پارامترهای این payload را داریم.



```
 0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
00 f8 08 ec f1 3f fa ff 87 49 15 90 57 71 62 0f  .....?...I..Wqb.
9b 4a ff 87 15 55 41 55 86 a1 0f b3 fa ff 06 49  .J...UAU.......I
[redacted]                 9b f8 01 f8 06 ec f4 ec f0 fb 10  ..@@C...........
45 d7 82 7c 62 43 53 a7 00 84 ae d9 b8 c5 09 d3  E..|bCS.........
fc 0c 63 61 6c 6c 2d 63 72 65 61 74 6f 72 fc 1c  ..call-creator..
[redacted]                    36 32 30 40 73 2e  [redacted]620@s
77 68 61 74 73 61 70 70 2e 6e 65 74 f8 06 f8 05  whatsapp.net....
0c ed 00 ff 02 80 00 d7 ec fb f8 05 0c ed 00 ff  ................
83 16 00 0f d7 ec fb f8 03 fc 03 6e 65 74 fc 06  ...........net..
6d 65 64 69 75 6d ff 81 3f f8 04 fc 0a 63 61 70  medium..?....cap
61 62 69 6c 69 74 79 fc 03 76 65 72 d3 fc 05 01  ability..ver....
03 ff 8b 1f f8 03 fc 06 65 6e 63 6f 70 74 fc 06  ........encopt..
6b 65 79 67 65 6e ff 81 2f f8 06 d7 c6 ff 81 2f  keygen../....../
b5 7e fc c9 33 08 e4 0c 12 21 05 60 ad 6a 75 00  .~..3....!.`.ju.
a2 59 77 fb 73 d9 a8 0b 8d e4 c2 19 98 b5 eb 30  .Yw.s..........0
d8 27 a8 7f ed 83 97 b7 9d 9c 76 1a 21 05 de 7a  .'........v.!..z
3d 1f a0 98 a7 5f 45 dc a4 6b 21 ad 54 ef 78 d4  =...._E..k!.T.x.
47 b1 30 d2 da 57 cc dd ec b8 9a b2 11 3e 22 72  G.0..W.......>"r
33 0a 21 05 b5 e1 72 2f 10 4c 5c 08 8d 7a fb e5  3.!...r/.L\..z..
20 65 a2 a0 b4 b9 51 54 27 54 1d 1e e0 61 08 10   e....QT'T...a..
7e 4a 9c 4c 10 23 18 00 22 40 fb 69 1b 80 c1 e6  ~J.L.#.."@.i....
12 09 48 f1 9f 63 fc 2c 05 96 d9 71 6e 62 2f 5d  ..H..c.,...qnb/]
41 d4 0b c6 44 b8 16 79 df 66 2c ec fc 0d 64 50  A...D..y.f,...dP
3f d0 02 2f 67 f1 2d 33 ef 64 70 dd 8f 13 1b 13  ?../g.-3.dp.....
87 56 82 47 80 a8 1f b5 4d 80 25 48 61 ee a2 8d  .V.G....M.%Ha...
7c fe 28 a1 a7 f1 84 01 30 ff 8e e5 02           |.(.....0....
```

این پیام یک پیام XMPP است که شامل پیام سیگنال ساخته شده در قسمت بالاتر است. XMPP مخفف پروتکل پیام رسانی و حضور گسترده پذیر پروتکل ارتباطی یک میان نرم افزار پیام محور بر پایه XML است. چون این یک پیام XMPP است ما به دنبال کلاس ها یا توابعی هستیم که نام XMPP را دارند و با جستجوی این نام به کلاسی به نام XMPP binary coder می‌رسیم. این کلاس در خودش تابع سریالایز را دارد که یک پیام XMPP را بصورت باینری نمایش می‌دهد. با استفاده از Frida مانند قبل وردوی ها و خروجی های این تابع را بدست می آوریم.

```
]-XMPPBinaryCoder serialize :
]call from@**********۴۹'=s.whatsapp.net'
```

id’۱۰-۱۵۵۵۴۱۵۵۸۶’=
to@**********۴۹’=**s.whatsapp.net**’
]offer call-id۴۵’=**D7827C624353A70084AED9B8C509D3**’
call-creator@**********۴۹’=**s.whatsapp.net**’
]audio rate ’۸۰۰۰’=enc’=**opus**[’
]audio rate ’۱۶۰۰۰’=enc’=**opus**[’
]net medium [’۳’=
]capability ver ’۱’={5b}[
]encopt keygen[’۲’=
]enc v ’۲’=type’=**pkmsg** ’{201b}[
[
[
[compressed۰ :**x0**[

کلاس XMPP دارای فیلدهای متنوعی است مانند ID کسی که تماس می گیرد و ID کسی تماس از سمت آن را دریافت می‌کند را شامل می‌شود. نکته جالب توجه در این قسمت این است که ما می‌توانیم با تغییر دادن این کلاس باعث ایجاد یک تماس ساختگی از سمت کسی که تماس نگرفته است داشته باشیم.

اینکار را به سادگی می‌توانیم با تغییر دادن call creator و fram در پیام XMPP انجام دهیم و به جای آن jabbar ID فردی که میخواهیم تماس آن را جعل کنیم را وارد کنیم. پس در این قسمت مشخص کردیم که برای پردازش کلید اصلی یک پیام در تماس واتساپ، کلید اصلی را رمزگذاری کرده و سپس در بسته هایی به صورت پیام سیگنال نگهداری می‌کنیم که پس از آن به فرمت یک قطعه باینری XMPP در می آید. این قطعه XMPP شامل آیدی تماس، jabber ID تماس گیرنده و فردی که با او تماس گرفته می‌شود.

۳ . ۴ انتقال کلید اصلی به سمت مخاطب تماس

طبق گفته وایت پیپر واتساپ برای ارسال تماس ها از noise protocol frame work استفاده می کند، به همین دلیل ما به دنبال توابع یا کلاسهایی میگردیم که در آن نام noise وجود داشته باشد تا این توابع را تحلیل کنیم. با انجام این کار کلاسی با نام WANoiseStreamCipher می یابیم که برای رمزنگاری پیام هایی استفاده می‌شود که به سمت سرورهای واتساپ می‌فرستیم. این تابع دارای متدی با نام encrypt plane text

است و با بررسی آن توسط Frida متوجه می‌شویم که مقدار plane text پس از برقراری یک تماس همان مقدار XMPP است که در قسمت بالا ساخته شد و پس از آن این پیام دوباره توسط تابع mbedtls_gcm_crypt_and_tag تگ از کتابخانه mbed TLS رمزگذاری می‌شود.

و پس از آن تابع mbedtls_gcm_setkey خوانده می‌شود تا از noise pipe با AES-256-GCM از noise protocol framework را نیز در رمزنگاری خود استفاده کنیم. این تابع کلیدی به طول ۲۵۶ بایت می سازد. این کلید توسط noise pipe protocol ساخته می‌شود. خروجی تمامی صدا زده سده در این قسمت یک متن ساده (Plane text) رمزنگاری شده است. که اگر پیام های ثبت شده توسط Wireshark را بررسی کنیم متوجه می‌شویم که این متن توسط یک بسته TCP به سمت سرورهای واتساپ فرستاده می‌شود و این سرور ها پس از دریافت این پیام، این پیام را به سمت فردی که قرار است با او تماس گرفته شود (مخاطب تماس) می‌فرستد. در زیر این این بسته TCP را مشاهده می‌کنیم که توسط Wireshark ثبت شده است.

```
    TCP payload (346 bytes)
    [Reassembled PDU in frame: 117]
    TCP segment data (346 bytes)
0000  60 0f 1e d9 01 7a 06 ff  20 01 4d d1 48 00 00 00   `....z.. .M.H...
0010  94 2c 6a f1 16 0d e6 90  2a 03 28 80 f2 3f 00 c8   .,j..... *.(..?..
0020  fa ce b0 0c 00 00 72 60  e6 4d 14 66 e9 c6 c1 fc   ......r` .M.f....
0030  ac 3a ee ce 80 18 08 00  58 ae 00 00 01 01 08 0a   .:...... X.......
0040  2c 13 e9 7a 0f de 7e 6d  00 01 57 45 ca fc 83 8f   ,..z..~m ..WE....
0050  3c 9e 6e 62 00 e6 0a a4  72 d7 3d 5c eb 20 40 0a   <.nb.... r.=\. @.
0060  10 d8 80 46 40 85 2f 41  0f 28 53 bb 9d 32 f3 26   ...F@./A .(S..2.&
0070  f8 f7 d6 f6 c8 06 db 6b  d0 b7 06 1f 31 78 a7 2e   .......k ....1x..
0080  dc 28 9a 80 42 b7 fd 7b  93 46 5b 49 19 2f bd 7f   .(..B..{ .F[I./..
0090  31 02 bf 0e 4c cf f7 83  df b4 d4 ca 3e f0 64 fb   1...L... ....>.d.
00a0  94 ec b9 d2 d0 47 79 1a  bb 5f 3e 3d 78 e5 ed 46   .....Gy. ._>=x..F
00b0  51 c8 73 a8 aa 57 95 bf  d7 d0 87 0e 85 8d 8e 1c   Q.s..W.. ........
00c0  82 64 f4 8c b5 e6 e3 8f  15 64 e0 c6 e2 28 d2 41   .d...... .d...(.A
00d0  4a 05 86 39 ec 61 98 bf  15 42 d5 e9 ca f4 0e fe   J..9.a.. .B......
00e0  6e 50 66 af 0f 52 4f a2  87 d1 98 4a aa fd dc 41   nPf..RO. ...J...A
00f0  e9 42 e3 ef 1c 63 c3 58  cd 73 fe 23 1f 7a 74 57   .B...c.X .s.#.ztW
0100  63 fe e8 fe b6 14 84 7b  91 49 68 f6 a4 6e 53 93   c......{ .Ih..nS.
0110  f4 bf bd 43 c9 4d d2 9a  ac bf d7 20 6a a7 f4 89   ...C.M.. ... j...
0120  5f 2f 2b d6 df 80 29 34  a2 7d 67 a5 24 91 d9 dc   _/+...)4 .}g.$...
0130  9b 3a 24 1d 72 4d 38 b6  68 56 44 66 44 56 f3 18   .:$.rM8. hVDfDV..
0140  10 a3 cc c9 ee 6d 7d db  e3 c3 bd 38 04 5d 07 70   .....m}. ...8.].p
0150  ac 00 0c 09 9d 4f 05 7a  8f ed a4 23 38 4f e8 ff   .....O.z ...#8O..
0160  71 d0 af ef c7 00 4d e7  47 ed 9c 7d 52 6a 75 ff   q.....M. G..}Rju.
0170  46 32 b3 04 e3 44 40 35  54 9e e6 bc ab 79 1c 25   F2...D@5 T....y.%
0180  06 36 c5 e7 e2 1a 52 70  58 f8 af ec ff 53 60 17   .6....Rp X....S`.
0190  31 0d ab 92 81 d5 b3 b6  73 c7 17 05 3a 5b 29 bb   1....... s...:[).
01a0  ec ae                                             ..
```

۳ . ۵ رمزگذاری تماس

برای رمزنگاری تماس از تابع srtp_aes_icm_encrypt از کتابخانه libsrtp صدا زده می‌شود. این موضوع را به این دلیل می‌گوییم که با مشاهده رفتار برنامه در طول یک تماس و مشاهده کردن رفتار آن توسط Frida رشته "block index: %d" مشاهده می‌شود.

و اگر در کد گیتاب این تابع در زیر نگاه کنیم دقیقا این رشته آمده است.

رشته زیر خروجی تابع srtp_aes_icm_encrypt را نشان می‌دهد که یک بسته SRTP است که توسط این تابع رمزنگاری شده است.

```
80 78 00 1f 00 02 99 00 f5 d2 f5 cf 53 9e bc 95 66 b9 9b 2d 73 5d 7d
43 35 3f ca c9 66 a6 64 e0 8d 2c 14 07 ef d6 5d 07 b7 50 38 7c 7c f2
56 43 42 b6 c4 5a 22 3c 16 ff f3 aa 4f 91 d8 f2 c1 2d 08 3e 5f e6 5b
0e c5 61 10 b2 36 c8 54 51 88 38 04 f7 70 2a c5 f6 fa 9b 84 f2 0b b3
e0 7e 23 1e 21 8d 9a a4 32 f9 e2 21 cc 15 6c dc bd f0 fc d1 0f 58 58
71 1c 1d c9 04 67 45 7e 8d 50 a9 ea c6 ec f8 3d 16 f4 cd 36 01 0a a2
8c
```

۳ . ۶ تمامیت تماس

در کتابخانه libsrtp شیوه رمزگشایی یک پیام توسط تابع srtp_hmac_compute انجام می‌شود و اگر به کد گیتاب این تابع نگاه کنیم این رشته هارد کد شده "intermediate state: %s" را در آن خواهیم دید.

حال با استفاده از Frida و دنبال کردن این رشته کد می‌توانیم تابع srtp_hmac_compute را در طول اجرا بیابیم و آن را تحلیل کنیم. اگر به کد این تابع نگاه کنیم این تابع بصورت پارامترهای ورودی و خروجی این تابع به صورت زیر است.

```
static srtp_err_status_t srtp_hmac_compute(void *statev, const uint8_t *message,
int msg_octets, int tag_len, uint8_t *result)
```

حال ما با استفاده از FRIDA ورودی ها و خروجی های این تابع را تحلیل می‌کنیم. در زیر تمام چیزهایی که Frida به ما نشان می‌دهد را مشاهده می‌کنیم، که شامل پیام و tag_len هستند.

Attaching...
search srtp_hmac_compute in memory from: 0x1016380ac
found srtp_hmac_compute at: 0x10163b5f4

tag_len: **10**
message: **81 ca 00 07 fe 67 2e 32 56 14 89 75 c5 c0 39 4a d3 a0 cd 48 8c 4b 61 8a 78 32 a7 89 1e b7 71 26 80 00 00 01**tag_len: **4**
message: **00 00 00 00**tag_len: **10**
message: **81 d0 00 02 fe 67 2e 32 b5 6f 93 8e 80 00 00 02**tag_len: **4**
message: **00 00 00 00**tag_len: **4**
message: **00 00 00 00**tag_len: **4**
message: **00 00 00 00**tag_len: **4**
message: **00 00 00 00**tag_len: **10**
message: **81 ca 00 07 83 42 f3 44 81 78 9f f5 39 b1 23 50 48 19 e0 f1 61 5b b5 32 dc b3 10 08 e7 47 a8 4b 80 00 00 01**tag_len: **10**
message: **81 d0 00 02 83 42 f3 44 94 60 21 fe 80 00 00 02**tag_len: **4**
message: **00 00 00 00**tag_len: **4**
message: **00 00 00 00**tag_len: **10**
message: **81 c8 00 12 fe 67 2e 32 87 b7 69 f8 5a 27 4c 76 b4 29 f6 5d 59 26 de af bd e9 4c 8b f3 ff 48 e3 a9 7e 62 cf db 9c 8a 3d 34 50 48 f8 fc 0e 88 7a 17 eb 17 94 9f 3d 91 27 89 d5 cc bd 21 ea 01 39 27 e1 05 07 66 69 1f 68 08 53 1a 18 02 9e bc 50 ed 8e 40 3e 8a 7b d3 b6 19 e8 54 6f 6b 58 ac 4e e3 25 f5 c2 e8 1c 97 bb 46 f9 38 45 80 00 00 03**...

از آنجا که در طول این تحقیق ما از برنامه Frida در زمان اجرا استفاده می‌کنیم و در طول انجام تماس Frida هنوز در حال اجرا است نمی‌توانیم از حفاظت کامل این بسته های SRTP مطمئن شویم.

نتیجه گیری

اگر بخواهیم تمام مطالبی که تا به این لحظه را جمع بندی کنیم، میتوانیم موارد زیر را از نتیجه این تحقیقمان بدست آوریم که واتساپ برای تماس از کتابخانه های libsignal-protocol-c, libsrtp, PJSIP برای پیاده سازی پروتکل تماس خود بر روی اینترنت استفاده می‌کند. پس از آن این کلید را بعنوان کلید اصلی برای تماس در نظر می گیرد و این کلید برای ساختن بسته های SRTP در تماس استفاده می‌شود و این کلید را بوسیله الگوریتم AES-128-ICM می‌سازد.

پس از آن این کلید اصلی را از طریق پروتکل سیگنال رمزگذاری می‌کنیم، سپس آنها را در پیام های XMPP بسته بندی می‌کنیم و سپس از طریق پروتکل Noise pipe این بسته ها را رمزگذاری می‌کنیم و این بسته ها را بصورت TCP به سمت سرورهای واتساپ می‌فرستیم و سپس سرور این کلید اصلی را بصورت رمزگذاری شده به سمت مخاطب تماس می‌فرستند و مخاطب تماس متوجه این می‌شود فرد دیگری در حال تماس گرفتن با او است.

منابع

https://medium.com/@schirrmacher/analyzing-whatsapp-calls-176a9e776213

https://www.dialogic.com/generatedpdfs/bn-sbc/bordernet_sbc_3-8-1/SRTP-User-Guide-V1.pdf