Technical Analysis of Code-Signed “Blister” Malware Campaign (Part 1)

cloudsek.com/technical-analysis-of-code-signed-blister-malware-campaign-part-1

Anandeshwar Unnikrishnan

January 7, 2022

A new malware, dubbed “Blister,” by the Elastic Security team that identified it, is leveraging valid code-signing certificates in Windows systems, to avoid detection by antivirus software. The malware is named after one of its payloads, Blister, which further deploys second-stage payloads.

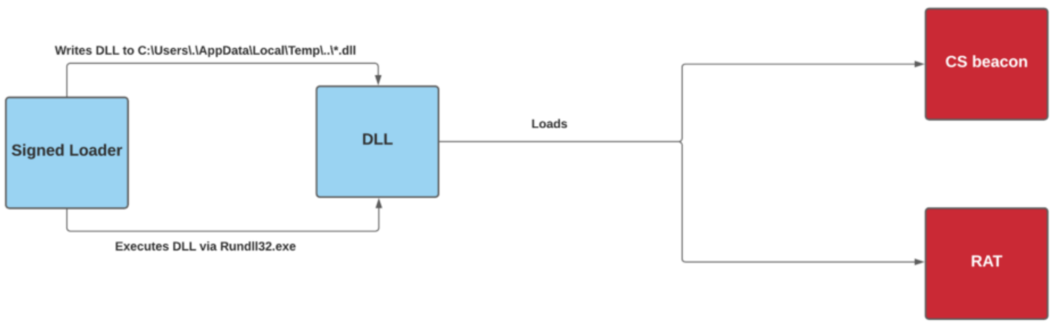
The threat actors orchestrating the Blister campaigns have been active since 15 September 2021, and have been using code-signing certificates that were validated on 23 August 2021. These certificates were issued by Sectigo to Blist LLC’s mail.ru email address. It is notable that mail.ru is a widely used Russian email service provider.

The malware masquerades malicious components as genuine executable files, due to which it has a low detection rate. Apart from using code-signing certificates, the threat actors are also leveraging other techniques, such as binding Blister to a legitimate library on the infected system, to stay under the radar.

Modus Operandi of the Blister Campaign

Threat actors are known to use code-signing to circumvent basic static security checks to compromise the victim systems. The Blister malware is no different in that it uses a Sectigo issued certificate to make the loader malware program look genuine to security products. It then deploys a Remote Access Trojan (RAT) on the target system to gain unauthorized access.

A *.dll* file is used as a second stage payload to execute the encoded RAT/ CobaltStrike beacon. Since the *.dll* file has no malicious traces there have been very few detections on VirusTotal. However, the loader uses *Rundll32.exe* to execute the *LaunchColorCpl* function exported by the malicious *.dll* file.

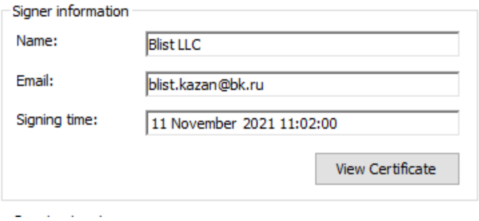

Writes DLL to C:\Users\.\AppData\Local\Temp\..*.dll
Signed Loader
DLL
Loads
CS beacon
RAT
Executes DLL via Rundll32.exe

Overview of the Blister malware campaign

Leveraging Code-Signing Certificates to Avoid Detection

The below image contains the details of the certificate to an entity called “Blist LLC”. It is common for cybercriminals to either steal code-signing certificates from compromised targets, or to use a front company to obtain the certificate, to sign the malware with.

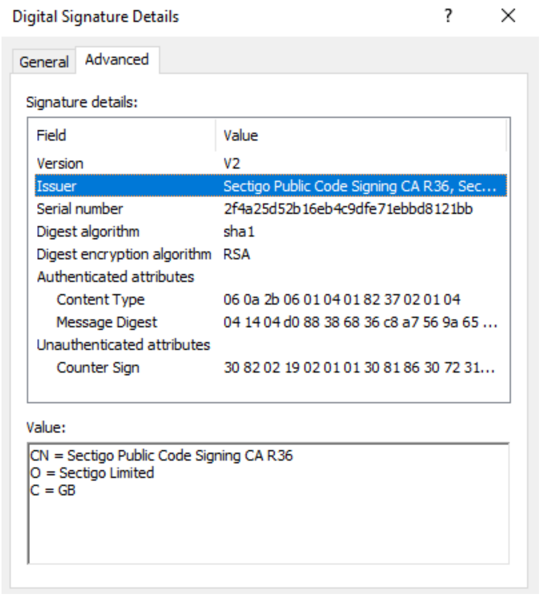
Sectigo has since revoked the certificate issued to the binary.



Signer information

Name:	Blist LLC
Email:	blist.kazan@bk.ru
Signing time:	11 November 2021 11:02:00

Certificate issued to Blist LLC



Signature details:

Field	Value
Version	V2
Issuer	Sectigo Public Code Signing CA R36, Sec...
Serial number	2f4a25d52b16eb4c9dfe71ebbd8121bb
Digest algorithm	sha1
Digest encryption algorithm	RSA
Authenticated attributes	
Content Type	06 0a 2b 06 01 04 01 82 37 02 01 04
Message Digest	04 14 04 d0 88 38 68 36 c8 a7 56 9a 65 ...
Unauthenticated attributes	
Counter Sign	30 82 02 19 02 01 01 30 81 86 30 72 31...

Value:

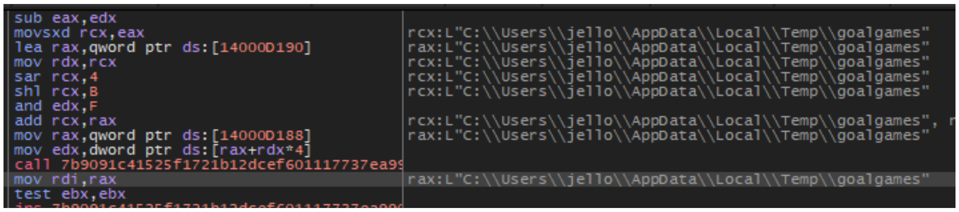
```
CN = Sectigo Public Code Signing CA R36
O = Sectigo Limited
C = GB
```

Certificate issued by Sectigo

First Stage of Infection

Overview of the Loader

- The loader writes a malicious *.dll* file in a directory created inside the user Temp folder.
- In one of the analysed samples, the malware created a folder named “goalgames” and inside it the loader dumped *holorui.dll*.
- The *.dll* houses the code for deploying the RAT to gain unauthorized access to the infected system.

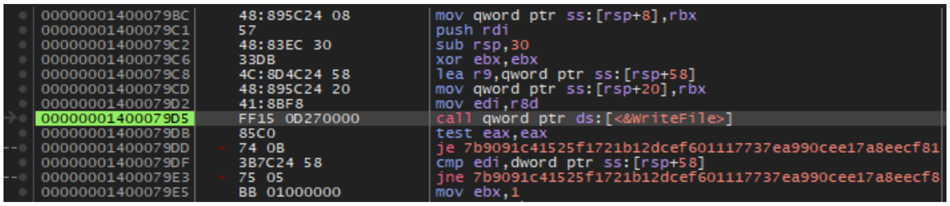


```
sub eax,edx
movsxd rcx,eax                    rcx:L"C:\\Users\\jello\\AppData\\Local\\Temp\\goalgames"
lea rax,qword ptr ds:[14000D190]  rax:L"C:\\Users\\jello\\AppData\\Local\\Temp\\goalgames"
mov rdx,rcx                       rcx:L"C:\\Users\\jello\\AppData\\Local\\Temp\\goalgames"
sar rcx,4                         rcx:L"C:\\Users\\jello\\AppData\\Local\\Temp\\goalgames"
shl rcx,B                         rcx:L"C:\\Users\\jello\\AppData\\Local\\Temp\\goalgames"
and edx,F
add rcx,rax                       rcx:L"C:\\Users\\jello\\AppData\\Local\\Temp\\goalgames", r
mov rax,qword ptr ds:[14000D188]  rax:L"C:\\Users\\jello\\AppData\\Local\\Temp\\goalgames"
mov edx,dword ptr ds:[rax+rdx*4]
call 7b9091c41525f1721b12dcef601117737ea99
mov rdi,rax                       rax:L"C:\\Users\\jello\\AppData\\Local\\Temp\\goalgames"
test ebx,ebx
```

The loader writes a .dll file in the user Temp folder

Step by Step Working of the Loader

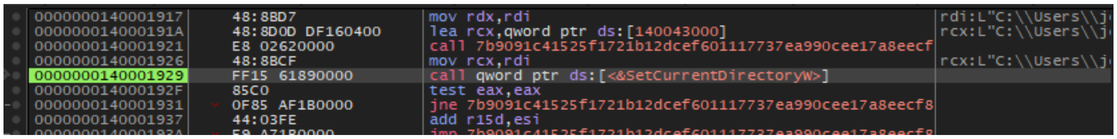
The Win32 API createDirectoryW is used to create a folder called “goalgames” in the path: *C:\Users\<user>\AppData\Local\Temp directory*. as shown below.



```
00000001400079BC   48:895C24 08     mov qword ptr ss:[rsp+8],rbx
00000001400079C1   57               push rdi
00000001400079C2   48:83EC 30       sub rsp,30
00000001400079C6   33DB             xor ebx,ebx
00000001400079C8   4C:8D4C24 58     lea r9,qword ptr ss:[rsp+58]
00000001400079CD   48:895C24 20     mov qword ptr ss:[rsp+20],rbx
00000001400079D2   41:8BF8          mov edi,r8d
00000001400079D5   FF15 0D270000    call qword ptr ds:[<&WriteFile>]
00000001400079DB   85C0             test eax,eax
00000001400079DD   74 0B            je 7b9091c41525f1721b12dcef601117737ea990cee17a8eecf81
00000001400079DF   3B7C24 58        cmp edi,dword ptr ss:[rsp+58]
00000001400079E3   75 05            jne 7b9091c41525f1721b12dcef601117737ea990cee17a8eecf8
00000001400079E5   BB 01000000      mov ebx,1
```

Using Win32 API createDirectoryW to create a folder in the user Temp folder

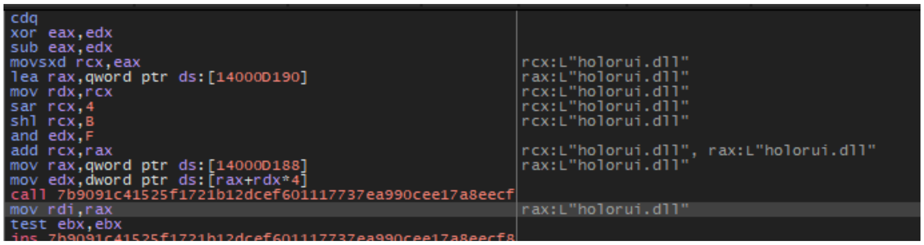
Before dumping the .dll, the loader sets the working directory to *C:\Users\<user>\AppData\Local\Temp\goalgames* via Win32 API *SetCurrentDirectoryW*.



```
0000000140001917   48:8BD7          mov rdx,rdi                                                rdi:L"C:\\Users\\j
000000014000191A   48:8D0D DF160400 lea rcx,qword ptr ds:[140043000]                           rcx:L"C:\\Users\\j
0000000140001921   E8 02620000      call 7b9091c41525f1721b12dcef601117737ea990cee17a8eecf
0000000140001926   48:8BCF          mov rcx,rdi                                                rcx:L"C:\\Users\\j
0000000140001929   FF15 61890000    call qword ptr ds:[<&SetCurrentDirectoryW>]
000000014000192F   85C0             test eax,eax
0000000140001931   0F85 AF1B0000    jne 7b9091c41525f1721b12dcef601117737ea990cee17a8eecf8
0000000140001937   44:03FE          add r15d,esi
```

Using Win32 API SetCurrentDirectoryW to set the working directory

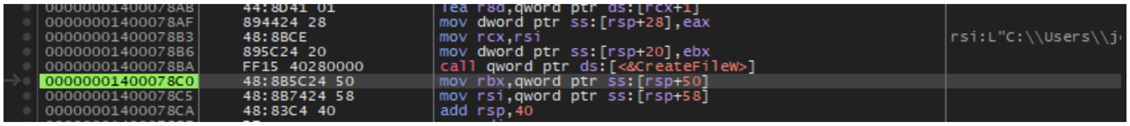
After setting the working directory, the malware resolves the filename for the *.dll* file to *holorui.dll* and stores it in the register RCX, to later pass it to Win32 API *CreateFileW*.



```
cdq
xor eax,edx
sub eax,edx
movsxd rcx,eax                                         rcx:L"holorui.dll"
lea rax,qword ptr ds:[14000D190]                       rax:L"holorui.dll"
mov rdx,rcx                                            rcx:L"holorui.dll"
sar rcx,4                                              rcx:L"holorui.dll"
shl rcx,B                                              rcx:L"holorui.dll"
and edx,F
add rcx,rax                                            rcx:L"holorui.dll", rax:L"holorui.dll"
mov rax,qword ptr ds:[14000D188]                       rax:L"holorui.dll"
mov edx,dword ptr ds:[rax+rdx*4]
call 7b9091c41525f1721b12dcef601117737ea990cee17a8eecf
mov rdi,rax                                            rax:L"holorui.dll"
test ebx,ebx
```

The malware resolves the filename for the .dll file to holorui.dll

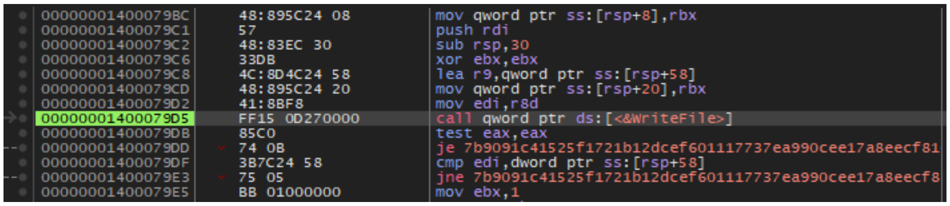
The file *C:\Users\<user>\AppData\Local\Temp\goalgames\holorui.dll* is created using the *CreateFileW* API.



```
00000001400078AB   44:8D41 01       lea r8d,qword ptr ds:[rcx+1]
00000001400078AF   894424 28        mov dword ptr ss:[rsp+28],eax
00000001400078B3   48:8BCE          mov rcx,rsi                                rsi:L"C:\\Users\\j
00000001400078B6   895C24 20        mov dword ptr ss:[rsp+20],ebx
00000001400078BA   FF15 40280000    call qword ptr ds:[<&CreateFileW>]
00000001400078C0   48:8B5C24 50     mov rbx,qword ptr ss:[rsp+50]
00000001400078C5   48:8B7424 58     mov rsi,qword ptr ss:[rsp+58]
00000001400078CA   48:83C4 40       add rsp,40
```

holorui.dll created using *CreateFileW* API

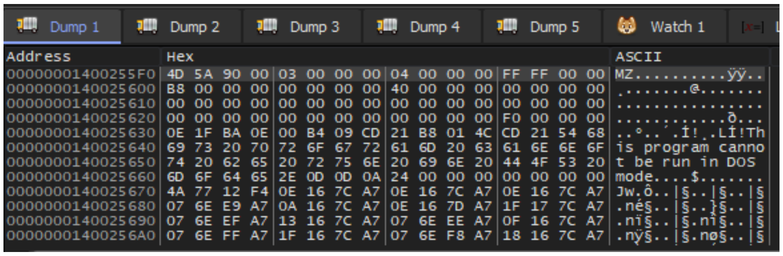
Once the file is created, the malware starts writing the content to the file by iteratively transferring bytes from the *.dll* payload in the loader. The Win32 API *WriteFile* is used to write contents into *holorui.dll*.



```
00000001400079BC   48:895C24 08     mov qword ptr ss:[rsp+8],rbx
00000001400079C1   57               push rdi
00000001400079C2   48:83EC 30       sub rsp,30
00000001400079C6   33DB             xor ebx,ebx
00000001400079C8   4C:8D4C24 58     lea r9,qword ptr ss:[rsp+58]
00000001400079CD   48:895C24 20     mov qword ptr ss:[rsp+20],rbx
00000001400079D2   41:8BF8          mov edi,r8d
00000001400079D5   FF15 0D270000    call qword ptr ds:[<&WriteFile>]
00000001400079DB   85C0             test eax,eax
00000001400079DD   74 0B            je 7b9091c41525f1721b12dcef601117737ea990cee17a8eecf81
00000001400079DF   3B7C24 58        cmp edi,dword ptr ss:[rsp+58]
00000001400079E3   75 05            jne 7b9091c41525f1721b12dcef601117737ea990cee17a8eecf8
00000001400079E5   BB 01000000      mov ebx,1
```

Win32 API WriteFile used to write contents into holorui.dll

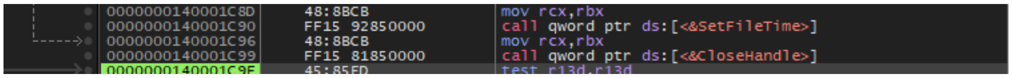
The malicious *.dll* is embedded in the initialized data segment of the PE executable of the loader and the bytes are transferred into *C:\Users\<user>\AppData\Local\Temp\goalgames\holorui.dll.*



```
Address          Hex                                              ASCII
00000001400255F0 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ..........ÿÿ..
0000000140025600 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ,.......@.......
0000000140025610 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0000000140025620 00 00 00 00 00 00 00 00 00 00 00 00 F0 00 00 00 ............ð...
0000000140025630 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 ..º..´.Í!¸.LÍ!Th
0000000140025640 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F is program canno
0000000140025650 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS 
0000000140025660 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 mode....$.......
0000000140025670 4A 77 12 F4 0E 16 7C A7 0E 16 7C A7 0E 16 7C A7 Jw.ô..|§..|§..|§
0000000140025680 07 6E E9 A7 0A 16 7C A7 0E 16 7D A7 1F 17 7C A7 .né§..|§..}§..|§
0000000140025690 07 6E EF A7 13 16 7C A7 07 6E EE A7 0F 16 7C A7 .nï§..|§.nî§..|§
00000001400256A0 07 6E FF A7 1F 16 7C A7 07 6E F8 A7 18 16 7C A7 .nÿ§..|§.nø§..|§
```

The MZ header of the embedded file

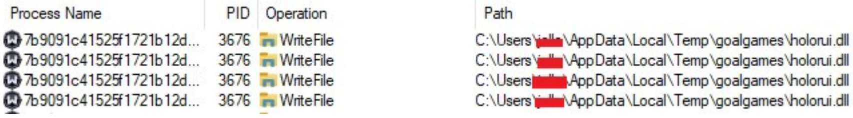
Upon closing the handle to the *holorui.dll* file, written on to the disk in the Temp directory, the malware finishes delivering the second stage payload. Then the file handles are closed by the malware.



```
000000014001C8D   48:8BCB         mov rcx,rbx
000000014001C90   FF15 92850000   call qword ptr ds:[<&SetFileTime>]
000000014001C96   48:8BCB         mov rcx,rbx
000000014001C99   FF15 81850000   call qword ptr ds:[<&CloseHandle>]
000000014001C9E   45:85ED         test r13d,r13d
```

File handles closed by the malware

The successful delivery of the malicious .dll can be confirmed by analyzing the interaction of the malware on the system.



```
Process Name              PID   Operation   Path
7b9091c41525f1721b12d...  3676  WriteFile   C:\Users\[redacted]\AppData\Local\Temp\goalgames\holorui.dll
7b9091c41525f1721b12d...  3676  WriteFile   C:\Users\[redacted]\AppData\Local\Temp\goalgames\holorui.dll
7b9091c41525f1721b12d...  3676  WriteFile   C:\Users[redacted]AppData\Local\Temp\goalgames\holorui.dll
7b9091c41525f1721b12d...  3676  WriteFile   C:\Users\[redacted]\AppData\Local\Temp\goalgames\holorui.dll
```

Successful delivery of the malicious .dll

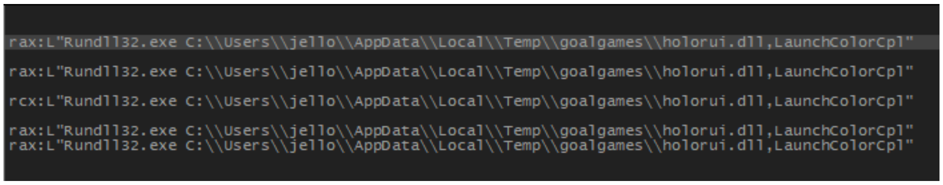
Based on analysing multiple signed loader samples, we have enumerated following distinct directory and payload names used within different samples from the same campaign:

- *C:\Users\<user>\AppData\Local\Temp\goalgames\holorui.dll*
- *C:\Users\<user>\AppData\Local\Temp\Framwork\axsssig.dll*
- *C:\Users\<user>\AppData\Local\Temp\oarimgamings\holorui.dll*
- *C:\Users\<user>\AppData\Local\Temp\guirtsframworks\Pasade.dll*

Note: The content inside the *.dll* is the same despite having different names

Second Stage of Infection

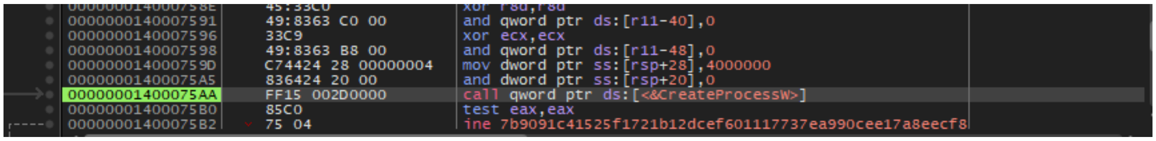
At the second stage of infection, the loader generates a command line to execute the function *LaunchColorCpl exported* from the *.dll,* via *Rundll32.exe* on the infected system.



```
rax:L"Rundll32.exe C:\\Users\\jello\\AppData\\Local\\Temp\\goalgames\\holorui.dll,LaunchColorCpl"

rax:L"Rundll32.exe C:\\Users\\jello\\AppData\\Local\\Temp\\goalgames\\holorui.dll,LaunchColorCpl"

rcx:L"Rundll32.exe C:\\Users\\jello\\AppData\\Local\\Temp\\goalgames\\holorui.dll,LaunchColorCpl"

rax:L"Rundll32.exe C:\\Users\\jello\\AppData\\Local\\Temp\\goalgames\\holorui.dll,LaunchColorCpl"
rax:L"Rundll32.exe C:\\Users\\jello\\AppData\\Local\\Temp\\goalgames\\holorui.dll,LaunchColorCpl"
```

Command line to execute the function LaunchColorCpl

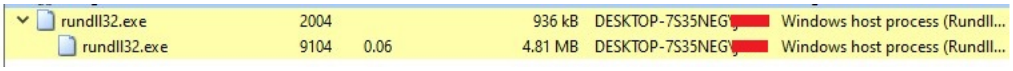
A new process is created with the above command line to spawn a *Rundll32* process via *CreateProcessW* Win32 API.



```
000000014000758E   45:33C0             xor r8d,r8d
0000000140007591   49:8363 C0 00       and qword ptr ds:[r11-40],0
0000000140007596   33C9                xor ecx,ecx
0000000140007598   49:8363 B8 00       and qword ptr ds:[r11-48],0
000000014000759D   C74424 28 00000004  mov dword ptr ss:[rsp+28],4000000
00000001400075A5   836424 20 00        and dword ptr ss:[rsp+20],0
00000001400075AA   FF15 002D0000       call qword ptr ds:[<&CreateProcessW>]
00000001400075B0   85C0                test eax,eax
00000001400075B2   75 04               jne 7b9091c41525f1721b12dcef601117737ea990cee17a8eecf8
```

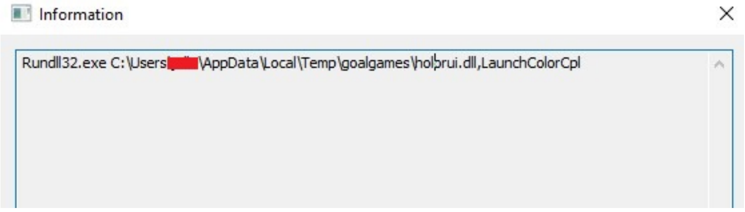
Spawning a *Rundll32* process via *CreateProcessW* Win32 API

The newly spawned *Rundll32.exe* process is listed in the process listing on the infected machine.



```
rundll32.exe        2004          936 kB   DESKTOP-7S35NEG[redacted]   Windows host process (Rundll...
  rundll32.exe      9104   0.06   4.81 MB  DESKTOP-7S35NEG[redacted]   Windows host process (Rundll...
```

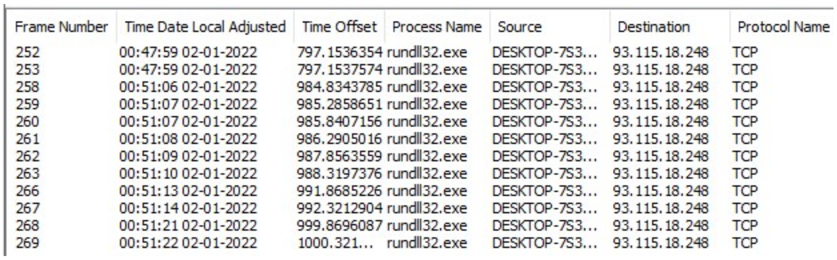
Newly spawned Rundll32.exe process



```
Rundll32.exe C:\Users\[redacted]\AppData\Local\Temp\goalgames\holbrui.dll,LaunchColorCpl
```

Command line confirmation for the newly spawned process

The final payload is executed by the *Rundll32.exe* process.



Frame Number	Time Date Local Adjusted	Time Offset	Process Name	Source	Destination	Protocol Name
252	00:47:59 02-01-2022	797.1536354	rundll32.exe	DESKTOP-7S3...	93.115.18.248	TCP
253	00:47:59 02-01-2022	797.1537574	rundll32.exe	DESKTOP-7S3...	93.115.18.248	TCP
258	00:51:06 02-01-2022	984.8343785	rundll32.exe	DESKTOP-7S3...	93.115.18.248	TCP
259	00:51:07 02-01-2022	985.2858651	rundll32.exe	DESKTOP-7S3...	93.115.18.248	TCP
260	00:51:07 02-01-2022	985.8407156	rundll32.exe	DESKTOP-7S3...	93.115.18.248	TCP
261	00:51:08 02-01-2022	986.2905016	rundll32.exe	DESKTOP-7S3...	93.115.18.248	TCP
262	00:51:09 02-01-2022	987.8563559	rundll32.exe	DESKTOP-7S3...	93.115.18.248	TCP
263	00:51:10 02-01-2022	988.3197376	rundll32.exe	DESKTOP-7S3...	93.115.18.248	TCP
266	00:51:13 02-01-2022	991.8685226	rundll32.exe	DESKTOP-7S3...	93.115.18.248	TCP
267	00:51:14 02-01-2022	992.3212904	rundll32.exe	DESKTOP-7S3...	93.115.18.248	TCP
268	00:51:21 02-01-2022	999.8696087	rundll32.exe	DESKTOP-7S3...	93.115.18.248	TCP
269	00:51:22 02-01-2022	1000.321...	rundll32.exe	DESKTOP-7S3...	93.115.18.248	TCP

Network activities between the infected host and the attacker C2

In the part 2 of this article we will cover the internal working of the .dll payload in detail.

Indicators of Compromise (IoCs)

FileHash-MD5

e6404260b4e42b7aa75bb0a96627ed3a	304921a919ab5228687a4932bb66fab9
db8827d0d7b2addc05719e407216da14	1b33c1f232b2ed68ac108519caa2d35f
755f50457416aeb7fee95a67abfea9fe	1896e6b20128e85a9851b94753eabbdf
6f76505a91c91c29238f0ed70b369417	a91ba8f4a339a98fa94e810831e83d96
5a7dea7aa86ccd600f5a97e3b53f7338	b8c9c560c6970a877a7ad359f37811d7
3efcd76417a185e48da71e22d230c547	

FileHash-SHA1

f8fa1ba14df6f8ab2b307ee0ce04054ea9d538c0	77b11cc7fc02f2ece71c380afbed82a39df9b8fa
f534e15bbc104cafab80f954ba30f12de87b0f48	72134bbf433c51d475412d16ff7abb4ce2b08110
d58e06727c551756cbee1fc6539929553a09878b	4800d1f8e6ebc489c6c8a1d3a1f99b8339cf0980
c039362e891b01040c20e75e16b02169c512aebd	21799d1d30344428697f3a186733b283a993ac16
bb69d5da32164813be5af29d31edc951a8f1f088	871e52778597185f98eb0a57127024bcd094cf07
a492b5e329b55d4a0f66217e5352ab56fabacad1	

FileHash-SHA256

fe7357d48906b68f094a81d19cc0ff93f56cc40454ac5f00e2e2d9c8ccdbc388	fa885e9ea1293552cb45a89e740426fa9c313225ff77ad1980dfe
f5104d0ead2f178711b1e23db3c16846de7d1a3ac04dbe09bacebb847775d76d	ed6910fd51d6373065a2f1d3580ad645f443bf0badc398aa7718
ed241c92f9bc969a160da2c4c0b006581fa54f9615646dd46467d24fe5526c7a	df8142e5cf897af65972041024ebe74c7915df0e18c6364c5fb9b
d54dfedda0efa36ed445d501845b61ab73c2102786be710ac19f697fc8d4ca5c	d0f934fd5d63a1524616bc13b51ce274539a8ead9b072e7f7fe1
cc31c124fc39025f5c3a410ed4108a56bb7c6e90b5819167a06800d02ef1f028	cb949ebe87c55c0ba6cf0525161e2e6670c1ae186ab83ce4604
ca09d9cd2f3cfcc06b33eff91d55602cb33a66ab3fd4f540b9212fce5ddae54a	c61d2ba1e001c137533cd7fb6b38fe71fee489d61dbcfea45c37
c0f3b27ae4f7db457a86a38244225cca35aa0960eb6a685ed350e99a36c32b61	bee3210360c5d0939c5d38b7b9f0c232cf9fbf93b46a19e53930
ba3a50930e7a144637faf88a98f2990a27532bfd20a93dc160eb2db4fbc17b58	afb77617a4ca637614c429440c78da438e190dd1ca24dc78483
af555d61becfcf0c13d4bc8ea7ab97dcdc6591f8c6bb892290898d28ebce1c5d	a486e836026e184f7d3f30eaa4308e2f0c381c070af1f525118a
a34821b50aadee0dd85c382c43f44dae1e5fef0febf2f7aed6abf3f3e21f7994	9bccc1862e3e5a6c89524f2d76144d121d0ee95b1b8ba5d0ffca
96bf7bd5f405d3b4c9a71bcd1060395f28f2466fdb91cafc6e261a31d41eb37a	9472d4cb393256a62a466f6601014e5cb04a71f115499c320dc
923b2f90749da76b997e1c7870ae3402aba875fdbdd64f79cbeba2f928884129	8e22cf159345852be585bc5a8e9af476b00bc91cdda98fd6a324
8ae2c205220c95f0f7e1f67030a9027822cc18e941b669e2a52a5dbb5af74bc9	8a414a40419e32282d33af3273ff73a596a7ac8738e9cdca6e7
863228efa55b54a8d03a87bb602a2e418856e0028ae409357454a6303b128224	84a67f191a93ee827c4829498d2cb1d27bdd9e47e136dc6652
81edf3a3b295b0189e54f79387e7df61250cc8eab4f1e8f42eb5042102df8f1f	7cd03b30cfeea07b5ea4c8976e6456cb65e09f6b8e7dcc68884
7b9091c41525f1721b12dcef601117737ea990cee17a8eecf81dcfb25ccb5a8f	6c6f808f9b19e1fab1c1b83dc99386f0ceee8593ddfd461ac047e
696f6274af4b9e8db4727269d43c83c350694bd1ef4bd5ccdc0806b1f014568a	56ca9ea3f7870561ed3c6387daf495404ed3827f212472501d2
5651e8a8e6f9c63c4c1162efadfcb4cdd9ad634c5e00a5ab03259fcdeaa225ac	516cac58a6bfec5b9c214b6bba0b724961148199d32fb42c01b
4fe551bcea5e07879ec84a7f1cea1036cfd0a3b03151403542cab6bd8541f8e5	44e5770751679f178f90ef7bd57e8e4ccfb6051767d8e906708c
3c7480998ade344b74e956f7d3a3f1a989aaf43446163a62f0a8ed34b0c010d0	359ffa33784cb357ddabc42be1dcb9854ddb113fd8d6caf3bf039
2d049f7658a8dccd930f7010b32ed1bc9a5cc0f8109b511ca2a77a2104301369	294c710f4074b37ade714c83b6b7bf722a46aef61c02ba6543de
25a0d6a839c4dc708dcdd1ef9395570cc86d54d4725b7daf56964017f66be3c1	216cb4f2caeaf59f297f72f7f271b084637e5087d59411ac77ddd
1a10a07413115c254cb7a5c4f63ff525e64adfe8bb60acef946bb7656b7a2b3d	17ea84d547e97a030d2b02ac2eaa9763ffb4f96f6c54659533a2
00eb2f75822abeb2e222d007bdec464bfbc3934b8be12983cc898b37c6ace081	0a7778cf6f9a1bd894e89f282f2e40f9d6c9cd4b72be97328e681

Domains

- discountshadesdirect.com
- domain clippershipintl.com
- domain bimelectrical.com

IPv4

- 93.115.18.248
- 188.68.221.203
- 185.170.213.186

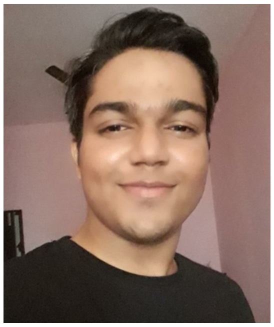
Signed loaders

- ed6910fd51d6373065a2f1d3580ad645f443bf0badc398aa77185324b0284db8
- cb949ebe87c55c0ba6cf0525161e2e6670c1ae186ab83ce46047446e9753a926
- 7b9091c41525f1721b12dcef601117737ea990cee17a8eecf81dcfb25ccb5a8f
- 84a67f191a93ee827c4829498d2cb1d27bdd9e47e136dc6652a5414dab440b74
- cc31c124fc39025f5c3a410ed4108a56bb7c6e90b5819167a06800d02ef1f028
- 9472d4cb393256a62a466f6601014e5cb04a71f115499c320dc615245c7594d4
- 4fe551bcea5e07879ec84a7f1cea1036cfd0a3b03151403542cab6bd8541f8e5
- 1a10a07413115c254cb7a5c4f63ff525e64adfe8bb60acef946bb7656b7a2b3d
- 9bccc1862e3e5a6c89524f2d76144d121d0ee95b1b8ba5d0ffcaa23025318a60
- 8a414a40419e32282d33af3273ff73a596a7ac8738e9cdca6e7db0e41c1a7658
- 923b2f90749da76b997e1c7870ae3402aba875fdbdd64f79cbeba2f928884129
- ed241c92f9bc969a160da2c4c0b006581fa54f9615646dd46467d24fe5526c7a
- 294c710f4074b37ade714c83b6b7bf722a46aef61c02ba6543de5d59edc97b60

DLL

BE7E259D5992180EADFE3F4F3AB1A5DECC6A394DF60C7170550B3D222FCE5F19

Anandeshwar Unnikrishnan
Threat Intelligence Researcher , CloudSEK
Anandeshwar is a Threat Intelligence Researcher at CloudSEK. He is a strong advocate of offensive cybersecurity. He is fuelled by his passion for cyber threats in a global context. He dedicates much of his time on Try Hack Me/ Hack The Box/ Offensive Security Playground. He believes that “a strong mind starts with a strong body.” When he is not gymming, he finds time to nurture his passion for teaching. He also likes to travel and experience new cultures.



Deepanjli Paulraj
Lead Cyberintelligence Editor, CloudSEK
Total Posts: 3
Deepanjli is CloudSEK’s Lead Technical Content Writer and Editor. She is a pen wielding pedant with an insatiable appetite for books, Sudoku, and epistemology. She works on any and all content at CloudSEK, which includes blogs, reports, product documentation, and everything in between.