

RESEARCH**MysterySnail attacks with Windows zero-day**12 OCT 2021  6 minute read[Table of Contents](#)**Executive Summary**

In late August and early September 2021, Kaspersky technologies detected attacks with the use of an elevation of privilege exploit on multiple Microsoft Windows servers. The exploit had numerous debug strings from an older, publicly known exploit for vulnerability [CVE-2016-3309](#), but closer analysis revealed that it was a zero-day. We discovered that it was using a previously unknown vulnerability in the Win32k driver and exploitation relies heavily on a technique to leak the base addresses of kernel modules. We promptly reported these findings to Microsoft. The information disclosure portion of the exploit chain was identified as not bypassing a security boundary, and was therefore not fixed. Microsoft assigned [CVE-2021-40449](#) to the use-after-free vulnerability in the Win32k kernel driver and it was patched on October 12, 2021, as a part of the October Patch Tuesday.

Besides finding the zero-day in the wild, we analyzed the malware payload used along with the zero-day exploit, and found that variants of the malware were detected in widespread espionage campaigns against IT companies, military/defense contractors, and diplomatic entities.

We are calling this cluster of activity MysterySnail. Code similarity and re-use of C2 infrastructure we discovered allowed us to connect these attacks with the actor known as IronHusky and Chinese-speaking APT activity dating back to 2012.

Elevation of privilege exploit

The discovered exploit is written to support the following Windows products:

- Microsoft Windows Vista
- Microsoft Windows 7
- Microsoft Windows 8
- Microsoft Windows 8.1
- Microsoft Windows Server 2008
- Microsoft Windows Server 2008 R2
- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2
- Microsoft Windows 10 (build 14393)
- Microsoft Windows Server 2016 (build 14393)
- Microsoft Windows 10 (build 17763)
- Microsoft Windows Server 2019 (build 17763)

The list of supported products and supported Windows 10 build numbers, explicit declaration of server OSs and the fact that exploits were only discovered in attacks on servers, all lead us to believe the exploit was developed and advertised as a solution to elevate privileges on servers.

CVE-2021-40449 is a use-after-free vulnerability in Win32k's NtGdiResetDC function. As with many other Win32k vulnerabilities, the root cause of this vulnerability lies in the ability to set user-mode callbacks and execute unexpected API functions during execution of those callbacks. The CVE-2021-40449 is triggered when the function ResetDC is executed a second time for the same handle during execution of its own callback. The exploitation process for this vulnerability is as follows:

- 1** A user-mode call to ResetDC executes syscall NtGdiResetDC and its inner function GreResetDCInternal. This function gets a pointer to a PDC object, and then performs a call to function hdcOpenDCW.
- 2** Function hdcOpenDCW performs a user-mode callback and it can be used to execute ResetDC for the same handle a second time.
- 3** If an exploit executes ResetDC during a callback, NtGdiResetDC and GreResetDCInternal are executed again for the same DC.
- 4** If an exploit ignores all the callbacks during the second call to GreResetDCInternal, this function will be executed as intended. It will create a new DC and get rid of the old one (the PDC object is destroyed).
- 5** In the callback, after the second ResetDC call has completed, the exploit can reclaim the freed memory of the PDC object and finish the execution of the callback.
- 6** After execution of the callback, function hdcOpenDCW returns to GreResetDCInternal, but the pointer retrieved in step (1) is now a dangling pointer – it points to the memory of the previously destroyed PDC object.

7 In the late stage of GreResetDCInternal execution, a malformed PDC object can be used to perform a call to an arbitrary kernel function with controlled parameters.

In the discovered exploit attackers are able to achieve the desired state of memory with the use of GDI palette objects and use a single call to a kernel function to build a primitive for reading and writing kernel memory. This step is easily accomplished, because the exploit process is running with Medium IL and therefore it's possible to use publicly known techniques to leak kernel addresses of currently loaded drivers/kernel modules. In our opinion, it would be preferable if the Medium IL processes had limited access to such functions as NtQuerySystemInformation or [EnumDeviceDrivers](#).













MysterySnail RAT

Our deep dive into the MysterySnail RAT family started with an analysis of a previously unknown remote shell-type Trojan that was intended to be executed by an elevation of privilege exploit. The sample which we analyzed was also [uploaded](#) to VT on August 10, 2021. The sample is very big – 8.29MB. One of the reasons for the file size is that it's statically compiled with the OpenSSL library and contains unused code and data belonging to that library. But the main reason for its size is the presence of two very large functions that do nothing but waste processor clock cycles. These functions also “use” randomly generated strings that are also present in a binary.

```
aNrmelc      db 'nrmelc',0          ; DATA XREF: MainFun1+D6⌵o
              ; MainFun2+CA⌵o
aUgbptznpwglott db 'ugBPTzNpwlOtTKIUb',0
              ; DATA XREF: MainFun1+F2⌵o
              ; MainFun2+E3⌵o
              align 10h
a0zphvcjovynx db 'ozPHVcJOVynX',0      ; DATA XREF: MainFun1+10C⌵o
              ; MainFun2+FD⌵o
              align 20h
aXvrkoyhmtsiuym db 'xvRK0YhmTSiuYMzPAsdYRFcxUBDUG',0
              ; DATA XREF: MainFun1+126⌵o
              ; MainFun2+117⌵o
              align 20h
aHcugesfponynhw db 'HCugeSFponYnHwYrMZq0vLdTOoBOYuYOr',0
              ; DATA XREF: MainFun1+140⌵o
```

Random strings used by anti-analysis functions

We assume these two functions are used as an AV-evasion technique for the purpose of anti-emulation. This theory is supported by the presence of other redundant logics and the presence of a relatively large number of exported functions while the real work is performed by only one of them.

Name	Address	Ordinal
 CheckBuf	00007FF8A65B2DD0	1
 ExeFile	00007FF8A65B3220	2
 Fcrypt32	00007FF8A65B2EF0	3
 Fprintf	00007FF8A65B2E70	4
 Fun_main	00007FF8A65B2FA0	5
 Fwrite	00007FF8A65B2EA0	6
 GetInfo	00007FF8A65B3310	7
 MainFun1	00007FF8A5FE88D0	8
 MainFun2	00007FF8A659B020	9
 TestRun	00007FF8A65B2FE0	10
 WorkFun	00007FF8A5FE8640	11
 DllEntryPoint	00007FF8A66DCDC4	6450236868 [...]

Names of exported functions; the actual business logic is executed from function "GetInfo"

The sample has two hardcoded URLs present in plain text – "www[.]disktest[.]com" and "www[.]runblerx[.]com". They are put into class variables for intended use, but remain unused; the real C2 address is decoded by a single byte xor – "http[.]ddspadus[.]com".

The malware enumerates the values under the "Software\Microsoft\Windows\CurrentVersion\Internet Settings\ProxyServer" registry key and uses them to request tunneling through a proxy server in case it fails to connect to the C2 directly.

The malware itself is not very sophisticated and has functionality similar to many other remote shells. But it still somehow stands out, with a relatively large number of implemented commands and extra capabilities like monitoring for inserted disk drives and the ability to act as a proxy.

Inbound and outbound commands have the same binary-based format that is provided below. All communication is encrypted with SSL.

Offset	Description
0	Size of additional data
4	Session ID
8	Command ID
0xC	Additional data

Format of communication commands

Before receiving any commands, the malware gathers and sends general information about the victim machine. This information includes:

- Computer name
- Current OEM code-page/default identifier

- Windows product name
- Local IP address
- Logged-in user name
- Campaign name

One interesting fact is that "Campaign name" by default is set to "windows". This name gets overwritten, but it might indicate there are versions of the same RAT compiled for other platforms.

In total, the RAT implements 20 commands. Their description and command IDs are provided in the table below.

Command ID	Description
1F4h	Launch interactive cmd.exe shell. Before launch cmd.exe is copied to the temp folder with a different name
1F5h	Spawn new process
1F6h	Spawn new process (console)
1F7h	Get existing disk drives and their type. This function also works in the background, checking for new drives
1F8h	Create (upload) new file. If a file exists, append data to it
1FAh	Get directory list
1FBh	Kill arbitrary process
1FFh	Delete file
202h	Read file. If the file is too big, async read operation can be stopped with cmd 20Ch.
205h	Re-connect
208h	Set sleep time (in ms)
209h	Shutdown network and exit
20Ah	Exit
20Bh	Kill interactive shell (created with cmd 1F4h)
20Ch	Terminate file reading operation (started with cmd 202h)
217h	No operation
21Bh	Open proxy'ed connection to provided host. Up to 50 simultaneous connections are

supported.

21Ch Send data to proxy'ed connection

21Eh Close all proxy connections

21Fh Close requested proxy connection

List of commands supported by the RAT

The analysis of the MysterySnail RAT helped us discover campaigns using other variants of the analyzed malware as well as study and document the code changes made to this tool over a six-month period. We provide more info about these variants and campaigns in our private report.

With the help of Kaspersky Threat Attribution Engine (KTAE) and the discovery of early variants of MysterySnail RAT we were able to find direct code and functionality overlap with the malware attributed to the IronHusky actor. We were also able to discover the re-use of C2 addresses used in attacks by the Chinese-speaking APT as far back as 2012. This discovery links IronHusky to some of the older known activities.

Kaspersky products detect the CVE-2021-40449 exploit and related malware with the verdicts:

- PDM:Exploit.Win32.Generic
- PDM:Trojan.Win32.Generic
- Trojan.Win64.Agent*

Kaspersky products detected these attacks with the help of the Behavioral Detection Engine and the Exploit Prevention component. CVE-2021-40449 is the latest addition to the long list of zero-days discovered in the wild with the help of our technologies. We will continue to improve defenses for our users by enhancing technologies and working with third-party vendors to patch vulnerabilities, making the internet more secure for everyone.

More information about these attacks and the actor behind them is available to customers of the Kaspersky Intelligence Reporting service. Contact: intelreports@kaspersky.com.

Kaspersky would like to thank Microsoft for their prompt analysis of the report and patches.

IoCs

www[.]disktest[.]com
www[.]runblerx[.]com
http[.]ddspadus[.]com

MD5 [e2f2d2832da0facbd716d6ad298073ca](#)

SHA1 [ecdec44d3ce31532d9831b139ea04bf48cde9090](#)

SHA256 [b7fb3623e31fb36fc3d3a4d99829e42910cad4da4fa7429a2d99a838e004366e](#)