

Check Point Research Unveil “Rorschach”

Previously Unseen,
Fastest Ever Ransomware



RORSCHACH – A NEW SOPHISTICATED AND FAST RANSOMWARE

April 4, 2023

Research by: Jiri Vinopal, Dennis Yarizadeh and Gil Gekker

Key Findings:

- Check Point Research (CPR) and Check Point Incident Response Team (CPIRT) encountered a previously unnamed ransomware strain, we dubbed **Rorschach**, deployed against a US-based company.
- Rorschach ransomware appears to be unique, sharing no overlaps that could easily attribute it to any known ransomware strain. In addition, it does not bear any kind of branding which is a common practice among ransomware groups.
- The ransomware is partly autonomous, carrying out tasks that are usually manually performed during enterprise-wide ransomware deployment, such as creating a domain group policy (GPO). In the past, similar functionality was linked to [LockBit 2.0](#).
- The ransomware is highly customizable and contains technically unique features, such as the use of direct syscalls, rarely observed in ransomware. Moreover, due to different implementation methods, **Rorschach** is one of the fastest ransomware observed, by the speed of encryption.
- The ransomware was deployed using DLL side-loading of a Cortex XDR Dump Service Tool, a signed commercial security product, a loading method which is not commonly used to load ransomware. The vulnerability was properly reported to Palo Alto Networks.

Introduction

While responding to a ransomware case against a US-based company, the CPIRT recently came across a unique ransomware strain deployed using a signed component of a commercial security product. Unlike other ransomware cases, the threat actor did not hide behind any alias and appears to have no affiliation to any of the known ransomware groups. Those two facts, rarities in the ransomware ecosystem, piqued CPR interest and prompted us to thoroughly analyze the newly discovered malware.

Throughout its analysis, the new ransomware exhibited unique features. A behavioral analysis of the new ransomware suggests it is partly autonomous, spreading itself automatically when executed on a Domain Controller (DC), while it clears the event logs of the affected machines. In addition, it's extremely flexible, operating not only based on a built-in configuration but also on numerous optional arguments which allow it to change its behavior according to the operator's needs. While it seems to have taken inspiration from some of the most infamous ransomware families, it also contains unique functionalities, rarely seen among ransomware, such as the use of direct syscalls.

The ransomware note sent out to the victim was formatted similarly to Yanluowang ransomware notes, although other variants dropped a note that more closely resembled DarkSide ransomware notes (causing some to mistakenly refer to it as DarkSide). Each person who examined the ransomware saw something a little bit different, prompting us to name it after the famous psychological test – Rorschach Ransomware.

Execution Flow

As observed in the wild, Rorschach execution uses these three files:

- cy.exe** – Cortex XDR Dump Service Tool version 7.3.0.16740, abused to side-load **winutils.dll**
- winutils.dll** – Packed Rorschach loader and injector, used to decrypt and inject the ransomware.
- config.ini** – Encrypted Rorschach ransomware which contains all the logic and configuration.

Upon execution of **cy.exe**, due to DLL side-loading, the loader/injector **winutils.dll** is loaded into memory and runs in the context of **cy.exe**. The main Rorschach payload **config.ini** is subsequently loaded into memory as well, decrypted and injected into **notepad.exe**, where the ransomware logic begins.

En cliquant sur « Accepter tous les cookies », vous acceptez le stockage de cookies sur votre appareil pour améliorer la navigation sur le site, analyser son utilisation et contribuer à nos efforts de marketing.

[Paramètres des cookies](#)

[Tout refuser](#)

[Autoriser tous les cookies](#)

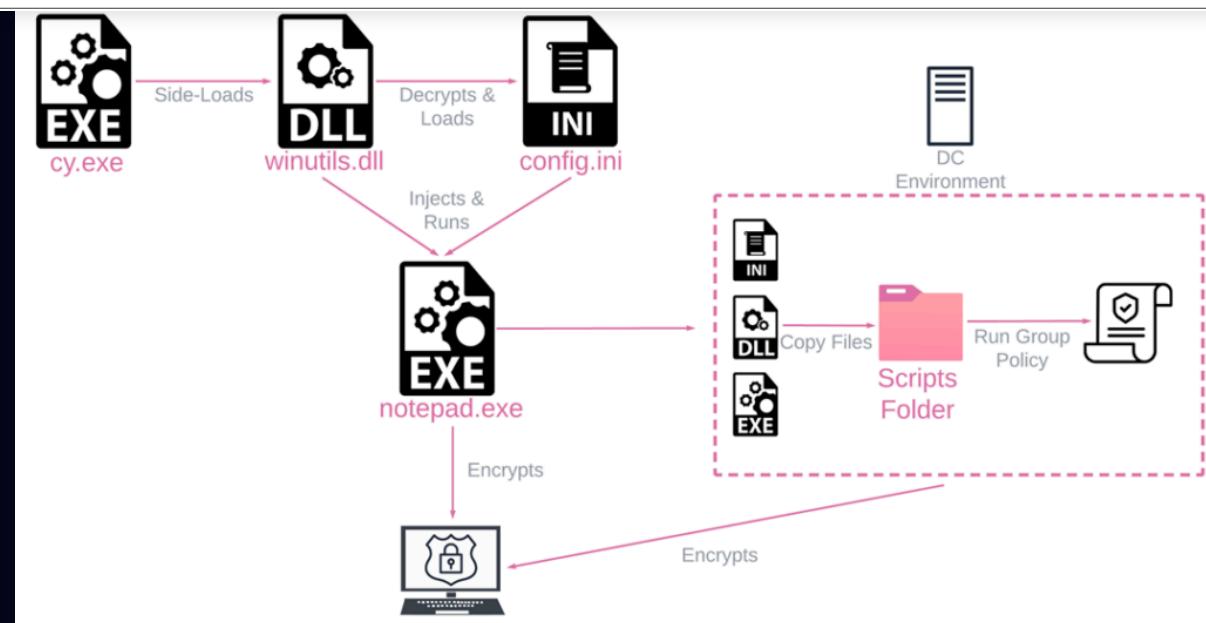


Figure 1 – Rorschach’s High Level Execution Flow on both endpoints and on Domain Controllers.

Security Solution Evasion

Rorschach spawns processes in an uncommon way, running them in SUSPEND mode and giving out falsified arguments to harden analysis and remediation efforts. The falsified argument, which consists of a repeating string of the digit 1 based on the length of the real argument, rewritten in memory and replaced with the real argument, resulting in a unique execution:

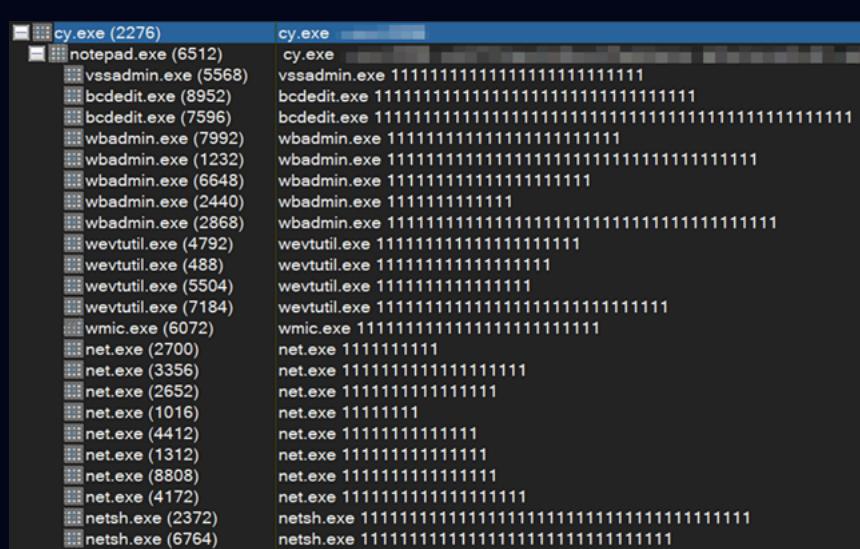


Figure 2 – Rorschach’s process tree spawns processes with falsified arguments.

The ransomware uses this technique to run the following operations:

- Attempt to stop a predefined list of services, using `net.exe stop`.
 - Delete shadow volumes and backups to harden recovery, using legitimate Windows tools such as `vssadmin.exe`, `bcdedit.exe`, `wmic.exe`, and `wbadmin.exe`
 - Run `wevutil.exe` to clear the following Windows event logs: Application, Security, System and Windows Powershell.
 - Disable the Windows firewall, using `netsh.exe`

Self-propagation

When executed on a Windows Domain Controller [DC], the ransomware automatically creates a Group Policy, spreading itself to other machines within the domain. Similar functionality was linked in the past to [LockBit 2.0](#), although the Rorschach Ransomware GPO deployment is carried out differently, as described below:

1. Rorschach copies its files into the scripts folder of the DC, and deletes them from the original location.
 2. Rorschach then creates a group policy (see Appendix C) that copies itself into the `%Public%` folder of all workstations in the domain.
 3. The ransomware creates another group policy in an attempt to kill a list of predefined list of processes. This is done by creating a schedule task invoking `taskkill.exe`.
 4. Finally, Rorschach creates another group policy that registers a scheduled task which runs immediately and upon user logon, to

Our colleagues in [AhnLab](#) published a more thorough behavioral analysis of another Rorschach variant which provides further details into

Ransomware Analytics

In addition to the ransomware's uncommon behavior described above, the Rorschach binary itself contains additional interesting

En cliquant sur « Accepter tous les cookies », vous acceptez le stockage de cookies sur votre appareil pour améliorer la navigation sur le site, analyser vos habitudes d'utilisation et vous proposer des publicités adaptées.

Security Solution Evasion

Although Rorschach is used solely for encrypting an environment, it incorporates an unusual technique to evade defense mechanisms. It makes direct system calls using the “syscall” instruction. While previously observed in other strains of malware, it’s quite startling to see this in ransomware.

The procedure involves utilizing the instruction itself, and it goes as follows:

1. The ransomware finds the relevant syscall numbers for NT APIs, mainly related to file manipulation.
2. Rorschach then stores the numbers in a table for future use.
3. When needed, it calls a stub routine that uses the number directly with the syscall instruction instead of using the NT API.

In other words, the malware first creates a syscall table for NT APIs used for file encryption:

```
NtSetInformationFile = GetSyscallNumber(&First, v25);
strcpy(&_Ptr[1], "NtQueryEaFile");
std::string(v32, &_Ptr[1]);
NtQueryEaFile_0 = GetSyscallNumber(&First, v26);
NtQueryEaFile = NtQueryEaFile_0;
if ( !NtCreateFile )
    return 0;
return NtReadFile
    && NtWriteFile
    && NtClose
    && NtQueryInformationFile
    && NtWaitForSingleObject
    && NtSetInformationFile
    && NtQueryEaFile_0;
```

Figure 3 – Creation of syscall table for certain NT APIs.

The end of the table is a section with the relevant syscall numbers:

Figure 4 – Section containing the syscall table.

The example below shows how the syscall numbers are used:

Figure 5 – Example use of direct syscall.

This obfuscated process is not required for the ransomware encryption logic, which suggests it was developed to bypass security solutions monitoring direct API calls.

Command Line Arguments

In addition to the hardcoded configuration, the ransomware comes with multiple built-in options, probably for the operators comfort. All of them are hidden, obfuscated, and not accessible without reverse-engineering the ransomware. This table contains some of the arguments that we discovered:

Argument	Example Parameter	Description
-run	=1234	Password needed to run the sample, possibly built on demand.
-nomutex	=1	Do not create a mutex, therefore do not insure that only a single instance is running.
-log	=1	Create log files.
-nodel	=0	Do not self-delete on execution.
-path	=“C:”	Encrypt only the following path

-we	= "C.exe"	Explicitly state the main executable.
-diskpart	=1	Run <code>diskpart.exe /s AppData_x.txt</code> that removes read-only volume attributes.
-nobk	=1	Do not change the wallpaper of the infected machine.
-thread	=4	Number of threads per CPU.
-at	=2023/03/24 05:04:20	Activation time (trigger time).
-nomail	=1	Do not create a ransom note.

This is only a partial list, with additional arguments suggesting networking capabilities, such as `listen`, `srv` and `hostfile`.

Example of how some of these arguments are used:

```
cy.exe --run=1234 --nomutex=0 --log=1 --node1=1 --path="C:\Myfolder" --full=1 --diskpart=1 --nobk=0
```

Language Based Protection

Before encrypting the target system, the sample runs two system checks that can halt its execution:

- It uses `GetSystemDefaultUILanguage` and `GetUserDefaultUILanguage` to determine what language the user is using.
- It exits if the return value is commonly used in CIS countries:

```
1. {
2.     0x042b: "Armenian_Armenia",
3.     0x042c: "Azeri_Latin",
4.     0x043f: "Kazakh",
5.     0x082c: "Azeri_Cyrillic",
6.     0x419: "Russian",
7.     0x422: "Ukrainian",
8.     0x423: "Belarusian",
9.     0x428: "Tajik",
10.    0x437: "Georgian",
11.    0x440: "Kyrgyz_Cyrillic",
12.    0x442: "Turkmen",
13.    0x443: "Uzbek_Latin",
14.    0x819: "Russian_Moldava",
15.    0x843: "Uzbek_Cyrillic"
16. }
```

Encryption Process

The Rorschach ransomware employs a highly effective and fast hybrid-cryptography scheme, which blends the `curve25519` and eSTREAM cipher `hc-128` algorithms for encryption purposes. This process only encrypts a specific portion of the original file content instead of the entire file. The WinAPI `CryptGenRandom` is utilized to generate cryptographically random bytes used as a per-victim private key. The shared secret is calculated through `curve25519`, using both the generated private key and a hardcoded public key. Finally, the computed SHA512 hash of the shared secret is used to construct the KEY and IV for the eSTREAM cipher `hc-128`.

Figure 6 – The Rorschach hybrid-cryptography scheme.

Analysis of Rorschach's encryption routine suggests not only the fast encryption scheme mentioned previously but also a highly effective implementation of thread scheduling via **I/O completion ports**. In addition, it appears that compiler optimization is prioritized for speed, with much of the code being inlined. All of these factors make us believe that we may be dealing with one of the fastest ransomware out there.

To verify our hypothesis, we conducted five separate encryption speed tests in a controlled environment (with 6 CPUs, 8192MB RAM, SSD, and 22000 files to be encrypted), limited to local drive encryption only. To provide a meaningful comparison with [other known fast ransomware](#), we compared Rorschach with the notorious LockBit v.3.

The result of the speed tests:

Ransomware	Average approximate time of encryption
LockBit v.3	7 minutes
Rorschach	4 minutes, 30 seconds

It turned out that we have a new speed demon in town. What's even more noteworthy is that the Rorschach ransomware is highly customizable. By adjusting the number of encryption threads via the command line argument `-thread`, it can achieve even faster times.

Technical Similarity to Other Ransomware

En cliquant sur « Accepter tous les cookies », vous acceptez le stockage de cookies sur votre appareil pour améliorer la navigation sur le site, analyser son utilisation et contribuer à nos efforts de marketing.

We discussed Rorschach's hybrid-cryptography scheme in detail above, but we suspect that this routine was borrowed from the [leaked source code](#) of **Babuk** ransomware. See the following code snippets as examples:

Figure 7 – Hybrid-cryptography scheme of Rorschach vs. Babuk.

Rorschach's inspiration from Babuk is evident in various routines, including those responsible for stopping processes and services. In fact, the code used to stop services through the service control manager appears to have been directly copied from Babuk's source code:

Figure 8 – Stopping predefined list of services – Rorschach vs. Babuk.

It is also worth noting that the list of **services** to be stopped in Rorschach's configuration is identical to that in the leaked Babuk source code. However, the list of **processes** to be stopped differs slightly, as Rorschach omits `notepad.exe`, which is used as a target for code injection.

Rorahsach takes inspiration from another ransomware strain: **LockBit**. First, the list of languages used to halt the malware is exactly the same list that was used in LockBit v2.0 (although the list is commonly used by many Russian speaking groups, and not just LockBit). However, the I/O Completion Ports method of thread scheduling is another component where Rorschach took some inspiration from LockBit. The final renaming of the encrypted machine files in Rorschach is implemented via `NtSetInformationFile` using `FileInformationClass FileRenameInformation`, just like in LockBit v2.0.

Figure 9 – Renaming of encrypted file using `NtSetInformationFile`.

As noted before, Rorschach's code is protected and obfuscated in a way that is unusual for ransomware, and is compiled with compiler optimization to favor speed and code inlining as much as possible. This makes finding similarities with other well-known ransomware families a real brain-buster. But we can still say that Rorschach took the best from the ransomware families with the highest reputation, and then added some unique features of its own.

Ransom Notes

As we noted, Rorschach does not exhibit any clear-cut overlaps with any of the known ransomware groups but does appear to draw inspiration from some of them.

We mentioned previously that Ahnlab reported a similar attack earlier this year. While it was carried out through different means, the ransomware described in the report triggers an almost identical execution flow. However, the resulting ransom note was completely different. The note was actually very similar to those issued by DarkSide, which probably led to this new ransomware being named "DarkSide," despite the group [being inactive](#) since May 2021.

The Rorschach variant we analyzed leaves a different ransom note based on the structure used by Yanlowang, another ransomware group:

Figure 10 – Ransom note from Rorschach.

Conclusion

Our analysis of Rorschach reveals the emergence of a new ransomware strain in the crimeware landscape. Its developers implemented new anti-analysis and defense evasion techniques to avoid detection and make it more difficult for security software and researchers to analyze and mitigate its effects. Additionally, Rorschach appears to have taken some of the 'best' features from some of the leading ransomwares leaked online, and integrated them all together. In addition to Rorschach's self-propagating capabilities, this raises the bar for ransom attacks. The operators and developers of the Rorschach ransomware remain unknown. They do not use branding, which is relatively rare in ransomware operations.

Our findings underscore the importance of maintaining strong cybersecurity measures to prevent ransomware attacks, as well as the need for continuous monitoring and analysis of new ransomware samples to stay ahead of evolving threats. As these attacks continue to grow in frequency and sophistication, it is essential for organizations to remain vigilant and proactive in their efforts to safeguard against these threats.

[Harmony Endpoint](#) provides runtime protection against ransomware with instant automated remediation, even in offline mode.

When running on a machine infected with the Rorschach ransomware, Harmony Endpoint Anti-ransomware detected the encryption process in different folders, including modifications made to Harmony Endpoint 'honeypot' files. It ran a ranking algorithm that provided a verdict identifying the process as a ransomware.

Samples/IOCs

Files

Name	Hash	Comments
cy.exe	2237ec542cdcd3eb656e86e43b461cd1	PA Cortex Dump Service Tool (benign file)
winutils.dll	4a03423c77fe2c8d979caca58a64ad6c	Loader and injector into notepad.exe
config.ini	6bd96d06cd7c4b084fe9346e55a81cf9	Encrypted ransomware payload

Appendix A – Services and processes terminated through GPO by Rorschach

The following services are stopped through a GPO issued by Rorschach, probably to prevent conflicting write orders to Database files (and thus preventing encryption):

```
SQLPBDMSSQLPBENGINE
MSSQLFDLauncher
SQLSERVERAGENT
MSSQLServerOLAPService
SSASTELEMETRY
SQLBrowser
SQL Server Distributed Replay Client
SQL Server Distributed Replay Controller
MsDtsServer150
SSISTELEMETRY150
SSISScaleOutMaster150
SSISScaleOutWorker150
MSSQLLaunchpad
SQLWriter
SQLTELEMETRY
MSSQLSERVER
```

The following processes are killed using a group policy (scheduled task) issued by Rorschach executing <C:\windows\system32\taskkill.exe>. Some are likely terminated to prevent write conflicts, and some are security solutions:

```
wxServer.exe
wxServerView.exe
sqlmang.exe
RAgui.exe
supervise.exe
Culture.exe
```

```
sqlbrowser.exe  
tomcat6.exe  
Sqlservr.exe
```

Appendix B – Hardcoded Rorschach configuration

The following is a list of services, hardcoded in its configuration, to be stopped via the service control manager:

```
AcronisAgent  
AcrSch2Svc  
backup  
BackupExecAgentAccelerator  
BackupExecAgentBrowser  
BackupExecDiveciMediaService  
BackupExecJobEngine  
BackupExecManagementService  
BackupExecRPCService  
BackupExecVSSProvider  
CAARCUpdateSvc  
CASAD2DWebSvc  
ccEvtMgr  
ccSetMgr  
DefWatch  
GxBlr  
GxCIMgr  
GxCVD  
GxFWD  
GxVss  
Intuit.QuickBooks.FCS  
memtas  
mepocs  
PDVFSService  
QBCFMonitorService  
QBFCService  
QBIDPService  
RTVscan  
SavRoam  
sophos  
sql  
stc_raw_agent  
svc$  
veeam  
VeeamDeploymentService  
VeeamNFSSvc  
VeeamTransportSvc  
VSNAPVSS  
vss  
YooBackup  
YooIT  
zhudongfangyu
```

The following is a hardcoded list of directories and files to be omitted from encryption:

```
.  
..  
#recycle  
$Recycle.Bin  
1_config.ini  
Ahnlab  
All Users  
AppData  
AUTOEXEC.BAT  
autoexec.bat  
autorun.inf  
begin.txt  
Boot  
boot.ini  
bootfont.bin  
bootmgfw.efi  
bootmgr  
bootmgr.efi  
bootsect.bak  
config.ini  
desktop.ini  
finish.txt  
Google  
iconcache.db  
Internet Explorer  
Mozilla  
Mozilla Firefox  
NETLOGON  
ntldr
```

Opera Software
Policies
Program Files
Program Files (x86)
ProgramData
scripts
SYSVOL
thumbs.db
Tor Browser
Windows
WINDOWS
Windows.old

The following is a list of process names that during Rorschach's execution these names are compared to those running on the machine and killed if matched. This is done through a combination of `CreateToolhelp32Snapshot`, `Process32FirstW`, `Process32NextW`, `OpenProcess`, and `TerminateProcess`. There is some overlap and redundancy to the list of services killed via the service control manager.

AcronisAgent
AcrSch2Svc
agntsvc.exe
BackExecRPCService
backup
BackupExecAgentAccelerator
BackupExecDiveciMediaService
BackupExecJobEngine
bedbg
CAARCUpdateSvc
ccEvtMgr
Culserver
dbeng50.exe
dbeng8
dbsnmp.exe
dbsrv12.exe
DefWatch
encsvc.exe
excel.exe
firefox.exe
infopath.exe
Intuit.QuickBooks.FCS
isqlplusvc.exe
memtas
mepocs
msaccess.exe
MSExchange
msftesql-Exchange
msmdsrv
mspub.exe
MSSQL
mydesktopqos.exe
mydesktopservice.exe
ocautoupds.exe
ocomm.exe
ocssd.exe
onenote.exe
oracle.exe
outlook.exe
PDVFSService
powerpt.exe
QBCFMonitorService
QBFCService
QBIDPService
SavRoam
sophos
sqbcoreservice.exe
sql.exe
sqladhlp
SQLADHLP
sqlagent
SQLAgent
SQLAgent\$SHAREPOINT
SQLBrowser
SQLWriter
steam.exe
synctime.exe
tbirdconfig.exe
thebat.exe
thunderbird.exe
tomcat6
veeam
VeeamDeploymentService
VeeamNFSSvc
VeeamTransportSvc

WSBExchange
xfssvccon.exe
YooBackup

Appendix C – Group Policies executed by Rorschach

Transferring its own files to each workstation:

```
1.  <Files clsid="{215B2E53-57CE-475c-80FE-9EEC14635851}">
2.    <File clsid="{50BE44C8-567A-4ed1-B1D0-9234FE1F38AF}" name="0305_winutils.dll"
   status="0305_winutils.dll" image="2" changed="2023-03-05 08:51:22" uid="{3F490769-A341-4220-
   90A3-51964B4A0C12}" bypassErrors="1">
3.      <Properties action="U">
4.        fromPath="\\**REDACTED**\sysvol\\**REDACTED**.local\scripts\winutils.dll"
   targetPath="%Public%\winutils.dll" readOnly="0" archive="1" hidden="0" suppress="0" />
5.      </File>
6.      <File clsid="{50BE44C8-567A-4ed1-B1D0-9234FE1F38AF}" name="0305_config.ini"
   status="0305_config.ini" image="2" changed="2023-03-05 08:51:22" uid="{F513F283-3C66-4C71-
   9B9B-4CE9BBFCEE1}" bypassErrors="1">
7.          <Properties action="U">
8.            fromPath="\\**REDACTED**.local\sysvol\\**REDACTED**.local\scripts\config.ini"
   targetPath="%Public%\config.ini" readOnly="0" archive="1" hidden="0" suppress="0" />
9.          </File>
10.         <File clsid="{50BE44C8-567A-4ed1-B1D0-9234FE1F38AF}" name="0305_cy.exe"
   status="0305_cy.exe" image="2" changed="2023-03-05 08:51:22" uid="{0A16D469-2648-4849-99C8-
   95D1B777D59A}" bypassErrors="1">
11.             <Properties action="U">
12.               fromPath="\\**REDACTED**.local\sysvol\\**REDACTED**.local\scripts\cy.exe"
   targetPath="%Public%\cy.exe" readOnly="0" archive="1" hidden="0" suppress="0" />
13.             </File>
14.           </Files>
```

Executing a scheduled task to run the attack:

```
1.  <TaskV2 clsid="{D8896631-B747-47a7-84A6-C155337F3BC8}" name="2_0305_cy.exe" image="2"
   changed="**REDACTED**" uid="{3772E17D-6354-4DF1-A73B-8868AC352B23}">
2.    <Properties action="U" name="2_0305_cy.exe" runAs="%LogonDomain%\%LogonUser%" 
   logonType="InteractiveToken">
3.      <Task version="1.2">
4.        <RegistrationInfo>
5.          <Author>**REDACTED**\Administrador</Author>
6.          <Description></Description>
7.        </RegistrationInfo>
8.        <Principals>
9.          <Principal id="Author">
10.            <UserId>%LogonDomain%\%LogonUser%</UserId>
11.            <LogonType>InteractiveToken</LogonType>
12.            <RunLevel>HighestAvailable</RunLevel>
13.          </Principal>
14.        </Principals>
15.        <Settings>
16.          <IdleSettings>
17.            <Duration>PT10M</Duration>
18.            <WaitTimeout>PT1H</WaitTimeout>
19.            <StopOnIdleEnd>false</StopOnIdleEnd>
20.            <RestartOnIdle>false</RestartOnIdle>
21.          </IdleSettings>
22.          <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
23.          <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>
24.          <StopIfGoingOnBatteries>false</StopIfGoingOnBatteries>
25.          <AllowHardTerminate>true</AllowHardTerminate>
26.          <AllowStartOnDemand>true</AllowStartOnDemand>
27.          <Enabled>true</Enabled>
28.          <Hidden>false</Hidden>
29.          <ExecutionTimeLimit>P3D</ExecutionTimeLimit>
30.          <Priority>7</Priority>
31.        </Settings>
32.        <Triggers>
33.          <RegistrationTrigger>
34.            <Enabled>true</Enabled>
35.          </RegistrationTrigger>
36.          <LogonTrigger>
37.            <Enabled>true</Enabled>
38.          </LogonTrigger>
39.        </Triggers>
40.        <Actions Context="Author">
41.          <Exec>
42.            <Command>%Public%\cy.exe</Command>
43.            <Arguments>--run=**REDACTED**</Arguments>
44.          </Exec>
45.        </Actions>
46.      </Task>
47.    </Properties>
48.  </TaskV2>
```



GO UP

BACK TO ALL POSTS



• • •



Publications

[Global cyber attack reports](#)

[Research publications](#)

[IPS advisories](#)

[Check point blog](#)

[Demos](#)

Tools

[Sandblast file analysis](#)

[ThreatCloud](#)

[Threat Intelligence](#)

[Zero day protection](#)

[Live threat map](#)

About Us

[Contact Us](#)

Let's get in touch

Subscribe for cpr blogs, news and more

[Subscribe Now](#)



© 1994-2024 Check Point Software Technologies LTD. All rights reserved.

Property of CheckPoint.com

[Privacy Policy](#)