Detecting DLL hijacking with machine learning: real-world cases

Introduction Our colleagues from the AI expertise center recently developed a machine-learning model that detects DLL-hijacking attacks. We then integrated this model into the Kaspersky Unified
Monitoring and Analysis Platform SIEM system. In a separate article , our colleagues shared how the model had been created and what success they had achieved in lab environments. Here, we focus on how it operates within Kaspersky SIEM, the preparation steps taken before its release, and some real-world incidents it has already helped us uncover. How the model works in Kaspersky SIEM The model s operation generally boils down to a
step-by-step check of all DLL libraries loaded by processes in the system, followed by validation in the Kaspersky Security
Network (KSN)
cloud. This approach allows local attributes (path, process name, and file hashes) to be combined with a global knowledge base and behavioral indicators, which significantly improves
detection quality and reduces the probability of false positives. The model can run in one of two modes: on a correlator or on a collector. A correlator is a SIEM component that performs event analysis and correlation based on predefined rules or algorithms. If
detection is configured on a correlator, the model checks events that have already triggered a rule. This reduces the volume of KSN queries and the model s response time. This is how it looks: A collector is a software or hardware component of a SIEM platform that collects and normalizes events from various sources, and then delivers these events to the platform s core. If
detection is configured on a collector, the model processes all events associated with various processes loading libraries, provided these events meet the following conditions: The path to the process file is known. The path to the library is known. The hashes of the file and the library are available. This method consumes more resources, and the model s response takes longer than it does on a correlator. However, it can be useful for retrospective threat
hunting because it allows you to check all events logged by Kaspersky SIEM. The model s workflow on a collector looks like this: It is important to note that the model is not limited to a binary malicious/non-malicious

assessment; it ranks its responses by confidence level. This allows it to be used as a flexible tool in SOC practice. Examples of possible verdicts: 0: data is being processed. 1: maliciousness not confirmed. This means the model currently does not consider the library malicious. 2: suspicious library. 3: maliciousness confirmed. A Kaspersky SIEM rule for detecting DLL hijacking would look like this: N.KL_AI_DLLHijackingCheckResult > 1 Embedding the model into the Kaspersky SIEM correlator automates the process of finding DLL-hijacking attacks, making it possible to detect them at scale without having to manually analyze hundreds or thousands of loaded libraries. Furthermore, when combined with correlation rules and telemetry sources, the model can be used not just as a standalone module but as part of a comprehensive defense against infrastructure attacks. Incidents detected during the pilot testing of the model in the MDR service Before being released, the model (as part of the Kaspersky SIEM platform) was tested in the MDR service, where it was trained to identify attacks on large datasets supplied by our telemetry. This step was necessary to ensure that detection works not only in lab settings but also in real client infrastructures. During the pilot testing, we verified the model s resilience to false positives and its ability to correctly classify behavior even in non-typical DLL-loading scenarios. As a result, several real-world incidents were successfully detected where attackers used one type of DLL hijacking the DLL Sideloading technique to gain persistence and execute their code in the system. Let us take a closer look at the three most interesting of these.

Incident 1. ToddyCat trying to launch Cobalt Strike disguised as a system library In one incident, the attackers successfully leveraged the vulnerability CVE-2021-27076 to exploit a SharePoint service that used IIS as a web server. They ran the following command:

```
c:\windows\system32\inetsrv\w3wp.exe -ap "SharePoint - 80" -v "v4.0" -l "webengine4.dll" -a \\.\pipe\iisipmd32ded38-e45b-423f-804d-34471928538b -h "C:\
```

inetpub\
temp\apppools\SharePoint - 80\SharePoint - 80.config" -w "" -m 0 After the exploitation, the IIS process created files that were later used to run malicious code via the DLL sideloading
technique (T1574.001 Hijack Execution Flow: DLL):

```
C:\
```

```
ProgramData
```

\SystemSettings
.exe

```
C:\
```

```
ProgramData
```

\SystemSettings.dll SystemSettings.dll is the name of a library associated with the Windows Settings application (SystemSettings
.exe). The original library contains code and data that the Settings application uses to manage and configure various system parameters. However, the library created by the attackers has malicious functionality and is only pretending to be a system library. Later, to establish persistence in the system and launch a DLL sideloading attack, a scheduled task was created, disguised as a Microsoft Edge browser

```
update
```

. It launches a SystemSettings
.exe file, which is located in the same directory as the malicious library:

```
Schtasks
```

/create /ru "SYSTEM" /tn "\Microsoft\Windows\Edge\Edgeupdates" /

```
sc
```

DAILY /tr "

```
C:\
```

```
ProgramData
```

\SystemSettings
.exe" /F The task is set to run daily. When the SystemSettings
.exe process is launched, it loads the malicious DLL. As this happened, the process and library data were sent to our model for analysis and detection of a potential attack. Example of a SystemSettings.dll load event with a DLL Hijacking module verdict in Kaspersky SIEM The resulting data helped our analysts highlight a suspicious DLL and analyze it in detail. The library was found to be a Cobalt Strike implant. After loading it, the SystemSettings
.exe process attempted to connect to the attackers command-and-

```
control
```

server. DNS
query
: connect-microsoft[.]com DNS
query
type: AAAA DNS response:
::ffff:8.219.1
[.]155; 8.219.1
[.]155:8443 After establishing a connection, the attackers began host reconnaissance to gather various data to develop their attack.
C:\
ProgramData
\SystemSettings
.exe
whoami
/priv hostname
reg
query
HKLM
\SOFTWARE\Microsoft\Cryptography /v MachineGuid
powershell
-c $psversiontable
dotnet
--version
systeminfo
reg
query
"HKEY_LOCAL_MACHINE\SOFTWARE\VMware, Inc.\VMware Drivers"
cmdkey
/list
REG
query
"
HKLM
\SYSTEM\CurrentControlSet\
Control
\Terminal Server\WinStations\RDP-Tcp" /v PortNumber
reg
query
"HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\Servers

```
netsh
wlan show profiles
netsh
wlan show interfaces set net localgroup administrators net user net user
administrator
ipconfig
/all net config workstation net view arp -a route
print
netstat -ano
tasklist
schtasks
/
query
/fo LIST /v net start net share net use
netsh
firewall show config
netsh
firewall show state net view /domain net time /domain net group "domain
admins" /domain net localgroup administrators /domain net group "domain
controllers" /domain net accounts /domain
nltest
/ domain_trusts
reg
query
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\
CurrentVersion\Run
reg
query
HKEY_CURRENT_USER\Software\Microsoft\Windows\
CurrentVersion\
RunOnce
reg
query
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\
CurrentVersion\Policies\
Explorer
\Run
reg
query
```

HKEY_LOCAL_MACHINE\Software\Wow6432Node\Microsoft\Windows\
CurrentVersion\Run
reg
query
HKEY_CURRENT_USER\Software\Wow6432Node\Microsoft\Windows\
CurrentVersion\
RunOnce
Based on the attackers TTPs, such as loading Cobalt Strike as a DLL , using the DLL sideloading
technique (1 , 2), and exploiting SharePoint, we can say with a high degree of confidence that the ToddyCat
APT group was behind the attack. Thanks to the prompt response of our model, we were able to respond in time and block this activity, preventing the attackers from causing damage to the organization. Another example was discovered by the model after a client was connected to MDR
monitoring: a legitimate system file located in an application folder attempted to load a suspicious library that was stored next to it.
C:\
Program Files\Chiniks\
SettingSyncHost.exe
C:\
Program Files\Chiniks\policymanager.dll E83F331BD1EC115524EBFF7043795BBE
The
SettingSyncHost.exe
file is a system host process for synchronizing settings between one user s different devices. Its 32-bit and 64-bit versions are usually located in
C:\
Windows\
System32
\ and
C:\
Windows\SysWOW64\, respectively. In this
incident, the file location differed from the normal one. Example of a policymanager.dll load event with a DLL Hijacking module verdict in Kaspersky SIEM Analysis of the library file loaded by this process showed that it was malware designed to steal information from browsers. Graph of policymanager.dll activity in a sandbox The file directly accesses browser files that contain user data.
C:\

Users\ \
AppData
\Local\Google\Chrome\User Data\Local State The library file is on the list of files used for DLL hijacking, as published in the HijackLibs project. The project contains a list of common processes and libraries employed in DLL-hijacking attacks, which can be used to detect these attacks. Another incident
discovered by our model occurred when a user connected a removable USB drive: Example of a Kaspersky SIEM event where a wsc.dll library was loaded from a USB drive, with a DLL Hijacking module verdict The connected drive s directory contained hidden folders with an identically named shortcut for each of them. The shortcuts had icons typically used for folders. Since file extensions were not shown by default on the drive, the user might have mistaken the shortcut for a folder and launched it. In turn, the shortcut opened the corresponding hidden folder and ran an
executable file using the following command: "%
comspec
%" /q /c "RECYCLER.BIN\1\CEFHelper
.exe [$DIGITS] [$DIGITS]" CEFHelper
.exe is a legitimate Avast Antivirus
executable that, through DLL sideloading, loaded the wsc.dll library, which is a malicious loader. Code snippet from the malicious file The loader opens a file named AvastAuth.dat, which contains an encrypted backdoor. The library reads the data from the file into memory, decrypts it, and executes it. After this, the backdoor attempts to connect to a
remote
command-and-
control
server. The library file, which contains the malicious loader, is on the list of known libraries used for DLL sideloading, as presented on the HijackLibs project website. Conclusion Integrating the model into the product provided the means of early and accurate
detection of DLL-hijacking attempts which previously might have gone unnoticed. Even during the pilot testing, the model proved its effectiveness by identifying several incidents using this
technique. Going forward, its accuracy will only increase as data accumulates and algorithms are updated in KSN, making this mechanism a reliable element of proactive protection for corporate systems.

IoC Legitimate files used for DLL hijacking

E0E092D4EFC15F25FD9C0923C52C33D6 loads SystemSettings.dll

09CD396C8F4B4989A83ED7A1F33F5503 loads policymanager.dll

A72036F635CECF0DCB1E9C6F49A8FA5B loads wsc.dll Malicious files

EA2882B05F8C11A285426F90859F23C6 SystemSettings.dll

E83F331BD1EC115524EBFF7043795BBE policymanager.dll

831252E7FA9BD6FA174715647EBCE516 wsc.dll Paths

C:\

ProgramData

\SystemSettings

.exe

C:\

ProgramData

\SystemSettings.dll

C:\

Program Files\Chiniks\

SettingSyncHost.exe

C:\

Program Files\Chiniks\policymanager.dll

D:\

RECYCLER.BIN\1\CEFHelper

.exe

D:\

RECYCLER.BIN\1\wsc.dll