New Yokai Side-loaded Backdoor Targets Thai Officials

By Nikhil Hegde, Jan Michael Alcantara

Published: 2024-12-13 · Archived: 2026-04-05 18:12:58 UTC

Summary

DLL side-loading is a popular technique used by threat actors to execute malicious payloads under the umbrella of a benign, usually legitimate, executable. This allows the threat actor to exploit whitelists in security products that exclude trusted executables from detection. Among others, this technique has been leveraged by APT41 to deploy DUSTTRAP and Daggerfly to deliver Nightdoor backdoor.

During threat hunting activities, the Netskope team discovered a legitimate iTop Data Recovery application side-loading a backdoor we named Yokai that, to the best of our knowledge, has not been publicly documented yet. In this blog we will analyze the infection chain and dive deep into the internals of the Yokai backdoor.

Decoy documents

During our threat hunting activities, we discovered a RAR file that contained two LNK shortcut files named in Thai, named กระทรวงยุติธรรมสหรัฐอเมริกา.pdf and ด่วนที่สุด ทางการสหรัฐอเมริกาขอความร่วมมือระหว่างประเทศในเรื่องทางอาญา.docx. Translated, both documents are called “United States Department of Justice.pdf” and “Urgently, United States authorities ask for international cooperation in criminal matters.docx” respectively.

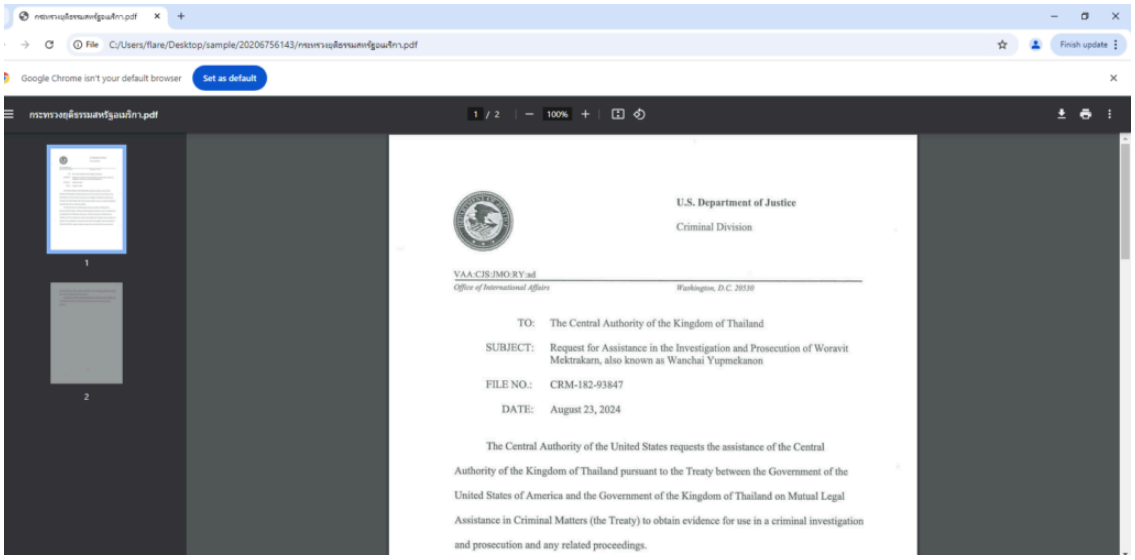
Clicking the shortcut files triggers the copying of content from an alternate data stream (ADS) named “file.exf” into decoy PDF and Word documents using esentutl.Esentutl is a Windows binary commonly abused to transfer malicious payload from alternate data streams.

```
PS C:\Users\Public\temp> get-item -path .\place.pdf.lnk -Stream *


PSPath        : Microsoft.PowerShell.Core\FileSystem::C:\Users\Public\temp\place.pdf.lnk::$DATA
PSParentPath  : Microsoft.PowerShell.Core\FileSystem::C:\Users\Public\temp
PSChildName   : place.pdf.lnk::$DATA
PSDrive       : C
PSProvider    : Microsoft.PowerShell.Core\FileSystem
PSIsContainer : False
FileName      : C:\Users\Public\temp\place.pdf.lnk
Stream        : :$DATA
Length        : 1459

PSPath        : Microsoft.PowerShell.Core\FileSystem::C:\Users\Public\temp\place.pdf.lnk:file.exf
PSParentPath  : Microsoft.PowerShell.Core\FileSystem::C:\Users\Public\temp
PSChildName   : place.pdf.lnk:file.exf
PSDrive       : C
PSProvider    : Microsoft.PowerShell.Core\FileSystem
PSIsContainer : False
FileName      : C:\Users\Public\temp\place.pdf.lnk
Stream        : file.exf
Length        : 243123
```

After the copying is complete, the shortcut files open the decoy documents, giving the impression that their sole purpose is to display harmless content.



Dropper emerges from an alternate data stream

The shortcut file with a DOCX extension contains two data streams: the “file.exf” which is used to create a decoy document, and “file.ext” which contains the malicious payload.

The malicious payload is copied to an executable named “file.exe” using esentutl. If successful, the command prompt launches “file.exe” using start command.

Upon execution of file.exe, it drops three files on the folder path C:\ProgramData\police. IdrInit.exe is a legitimate iTop Data Recovery application abused to side-load the malicious DLL ProductStatistics3.dll. IdrInit.exe is executed by file.exe. The file IdrInit.exe.data contains information sent by the C2. The contents will be described later in this blog.

Yokai Backdoor

Upon execution, IdrInit.exe side-loads the Yokai backdoor (ProductStatistics3.dll) and calls the “SetStatParam3” function from it. This function is simply a wrapper around the malicious “pfc_init” function.

In another variant of this backdoor, we found a PDB string "E:\Project\YK0163\src\YK0163_v2.0.0\YK0163\Release\EACore.pdb", which we used as an inspiration to name this backdoor, "Yokai".

Scheduled task and duplicate executions

If IdrInit.exe is executed as a non-admin user, Yokai creates a scheduled task, even if it already exists, named "MicrosoftTSUpdate" which executes IdrInit.exe every five minutes.

```
cmd.exe /c schtasks /create /f /sc MINUTE /MO 5 /tn "MicrosoftTSUpdate" /tr <path_to_IdrInit.exe>
```

If IdrInit.exe is executed as an admin user, it spawns a copy of itself. Each spawned process, in turn, spawns its own copy, and so on. If the number of spawned processes continuously increases, it could negatively impact system performance and raise likelihood of detection. Additionally, this behavior would be clearly visible if the user inspected the processes running on their system, further reducing the stealth aspect of the malware.

Mutex creation

Yokai looks for the presence of a mutex named “Mutex_Local_Windows_1547”. If it exists, control returns to IdrInit.exe and it terminates itself; otherwise, the mutex is created. Malware often uses mutexes as a mechanism to avoid duplicate executions on a host it has already compromised. It would have been a better design choice to first check for the existence of the mutex before creating a scheduled task or spawning processes, as described earlier.

Check-in with the C2

Yokai collects the hostname and username of the logged-in user. A string “2.2.0” was also found hardcoded and the PDB string mentioned earlier had the substring “2.0.0” in it. We believe this string indicates the version of the malware.

The data is then formatted into the following 156-byte block:

		Header				**Payload**	
Checksum (1 byte)	API-INDEX (4 bytes)	Sequence number (2 bytes)	Undetermined (1 byte)	Payload size (4 bytes)	Malware version (16 bytes)	Username (64 bytes)	Hostname (64 bytes)

The first byte serves as a checksum calculated over the remaining data. (The API-INDEX value will be described later in this blog.) The sequence number represents a 0-indexed count of the backdoor's communication interactions with the C2, incrementing by 1 with each new successful exchange.

The checksum is calculated via XOR operations, which, although unreliable, do provide basic error detection. This is calculated in the following manner:

```
def calculate_checksum(data):
    for i in range(1, len(data)):
        data[0] ^= data[i]
    # The first byte of the data is the checksum
    return data[0]
```

The data is then encrypted using a series of XOR operations with a hardcoded key, “ExtensionsWindow”. The algorithm is shown below:

```
def encrypt_data(plaintext, key):
    encrypted1 = b""
    for i in range(0, len(plaintext)):
        encrypted1 += (plaintext[i] ^ key[i & 0xF]).to_bytes(1, "little")
    encrypted2 = b""
    for i in range(1, len(encrypted1)+1):
        encrypted2 += (encrypted1[i-1] ^ key[i & 0xF]).to_bytes(1, "little")
    return encrypted2
```

The encrypted data is transmitted to the first available C2, which is one of the following in order:

- 122.155.28[.]155:80/page/index.php
- 154.90.47[.]77:80/

In another variant, we observed traffic going to port 443. The following table shows the fixed values for specific HTTP headers:

Header name	**Value**
Content-Type	application/x-www-form-urlencoded
User-Agent	WinHTTP Example/1.0
API-INDEX	0
Accept-Connect	0
Content-Length	156

The C2 response is encrypted in the same manner and with the same encryption key as before. It can be decrypted as follows:

```
def decrypt_data(encrypted, key):
    decrypted1 = b""
    for i in range(1, len(encrypted)+1):
        decrypted1 += (encrypted[i-1] ^ key[i & 0xF]).to_bytes(1, "little")
    decrypted2 = b""
    for i in range(0, len(decrypted1)):
        decrypted2 += (decrypted1[i] ^ key[i & 0xF]).to_bytes(1, "little")
    return decrypted2
```

The decrypted C2 response must satisfy checksum validation, be exactly 28 bytes in size, and have a value of 1 in the 4th byte. It has the following structure:

	Header			**Payload**	
Checksum (1 byte)	Sequence number (2 bytes)	Command code (1 byte)	Payload size (4 bytes)	API-INDEX (4 bytes)	New encryption key (16 bytes)

The value of “API-INDEX” sent by the C2 is written to a file named “IdrInit.exe.data” in the same directory as IdrInit.exe. This value is used for the “API-INDEX” HTTP header in subsequent C2 communications. While we do not know the intention behind this header, it could be a mechanism to uniquely identify a compromised host as determined by the C2. The C2 also provides a new encryption key to be used for encrypting data before exfiltration.

C2 command codes

Yokai now enters a loop of processing commands sent by the C2 and data exfiltration. Its traffic to the C2 has the following structure:

		Header			**Payload**
Checksum (1 byte)	API-INDEX (4 bytes)	Sequence number (2 bytes)	Undetermined (1 byte)	Payload size (4 bytes)	Exfiltrated data (Optional)

The following table shows the fixed values for specific HTTP headers:

Header name	**Value**
Content-Type	application/x-www-form-urlencoded
User-Agent	WinHTTP Example/1.0
API-INDEX	Previously sent by the C2
Accept-Connect	1

For example, in the snapshot below, the traffic on the left represents an empty payload being sent to the C2, indicating idle time, while the traffic on the right represents data exfiltration.

The C2 response has the following structure:

	Header			**Payload**
Checksum (1 byte)	Sequence number (2 bytes)	Command code (1 byte)	Payload size (4 bytes)	Shell commands to execute (Optional)

The sequence number sent by the C2 must match the sequence number previously sent by Yokai. Otherwise, it checks if the sequence number sent by the C2 is one behind the sequence number previously sent by Yokai. If so, it retransmits; otherwise, it restarts the entire C2 communication process.

The table below summarizes the supported command codes:

Command code	Function
1	Represents the C2 ACK to the check-in traffic first sent by Yokai. A new encryption key and API-INDEX value is also sent by the C2.
3	Spawns cmd[.]exe and attaches its STDIN and STDOUT to anonymous pipes.
4	Execute shell commands by writing to pipes associated with the previously spawned cmd[.]exe.
2	Do nothing
5	Undetermined

Conclusions

In this blog, we walked through an infection chain that leveraged a RAR file that contained two LNK files. While one of them had a decoy PDF document embedded in its alternate data stream, the other embedded a decoy DOCX document and a malware dropper executable. It dropped a legitimate iTop Data Recovery application along with the side-loaded Yokai backdoor, offering console access to the threat actor.

DLL side-loading continues to be a popular technique leveraged by threat actors. Netskope will continue to track the evolution of the Yokai backdoor.

Recommendations

Netskope recommends that organizations review their security policies to ensure that they are adequately protected against malware:

- Inspect all HTTP and HTTPS traffic, including all web and cloud traffic, to prevent systems from communicating with malicious domains and IP addresses. Netskope customers can configure their Netskope NG-SWG with a URL filtering policy to block known malicious domains, and a threat protection policy to inspect all web content to identify malicious content using a combination of signatures, threat intelligence, and machine learning.
- Use Remote Browser Isolation (RBI) technology to provide additional protection when there is a need to visit websites that fall into categories that can present higher risk, like Newly Observed and Newly Registered Domains.

Netskope Detection

- Netskope Threat Protection
  - RAR: Shortcut.Trojan.Pantera
  - LNK: Shortcut.Trojan.Pantera
  - Dropper:
    - Trojan.Generic.37082949

        - Gen.Detect.By.NSCloudSandbox.tr
    - Yokai Backdoor:
        - Win32.Trojan.Generic
        - Gen:Variant.Lazy.572161
        - Gen:Variant.Mikey.170188
        - Trojan.Generic.37080167
- Netskope Intrusion Prevention System
    - Yokai Backdoor C2 traffic:
        - SID 170111: MALWARE-CNC Win.Backdoor.Yokai variant C2 outbound traffic detected
        - SID 170112: MALWARE-CNC Win.Backdoor.Yokai variant C2 outbound traffic detected

MITRE ATT&CK Techniques

Tactic	Technique
TA0002: Execution	T1053: Scheduled Task/Job T1559: Inter-Process Communication
TA0005: Defense Evasion	T1564.004: Hide Artifacts: NTFS File Attributes T1036: Masquerading T1574.002: Hijack Execution Flow: DLL Side-Loading T1480.002: Execution Guardrails: Mutual Exclusion
TA0011: Command and Control	T1071.001: Application Layer Protocol: Web Protocols T1573.001: Encrypted Channel: Symmetric Cryptography

IOCs

All the IOCs and scripts related to this malware can be found in our GitHub repository.

Source: https://www.netskope.com/blog/new-yokai-side-loaded-backdoor-targets-thai-officials