Mustang Panda's Hodur : Vieux trucs, nouvelle variante de Korplug

By Alexandre Côté Cyr

Archived: 2026-04-05 22:36:46 UTC

Les chercheurs d'ESET ont découvert une campagne toujours en cours utilisant une variante Korplug non documentée auparavant, qu'ils ont nommée Hodur en raison de sa ressemblance avec le variant THOR précédemment documentée par l'Unité 42 en 2020. Dans la mythologie nordique, Hodur est le demi-frère aveugle de Thor, qui a été trompé par Loki pour tuer leur demi-frère Baldr.

Principales conclusions de cet article de blog :

- En mars 2022, cette campagne est toujours en cours et remonte au moins à août 2021.
- Les victimes connues comprennent des entités de recherche, des fournisseurs de services Internet et des missions diplomatiques européennes.
- La chaîne de compromission comprend des documents leurres qui sont fréquemment mis à jour et se rapportent à des événements en Europe.
- La campagne utilise un chargeur personnalisé pour exécuter une nouvelle variante de Korplug.
- Chaque étape du processus de déploiement utilise des techniques d'anti-analyse et d'obscurcissement du flux de contrôle, ce qui la distingue des autres campagnes.
- Les chercheurs d'ESET fournissent une analyse approfondie des capacités et des commandes de cette nouvelle variante.

Les victimes de cette campagne sont probablement attirées par des documents servant de tentatives d'hameçonnage instrumentalisant les récents événements survenus en Europe, tels que l'invasion de l'Ukraine par la Russie. Plus de trois millions d'habitants ont ainsi fui la guerre vers les pays voisins, ce qui a entraîné une crise sans précédent aux frontières de l'Ukraine. L'un des noms de fichiers liés à cette campagne est Situation at the EU borders with Ukraine.exe.

D'autres leurres de phishing mentionnent des restrictions de voyage COVID-19 mises à jour, une carte d'aide régionale approuvée pour la Grèce et un règlement du Parlement européen et du Conseil. Le dernier est un document réel disponible sur le site Web du Conseil européen. Cela montre que le groupe APT à l'origine de cette campagne suit l'actualité et est capable d'y réagir rapidement et avec succès.

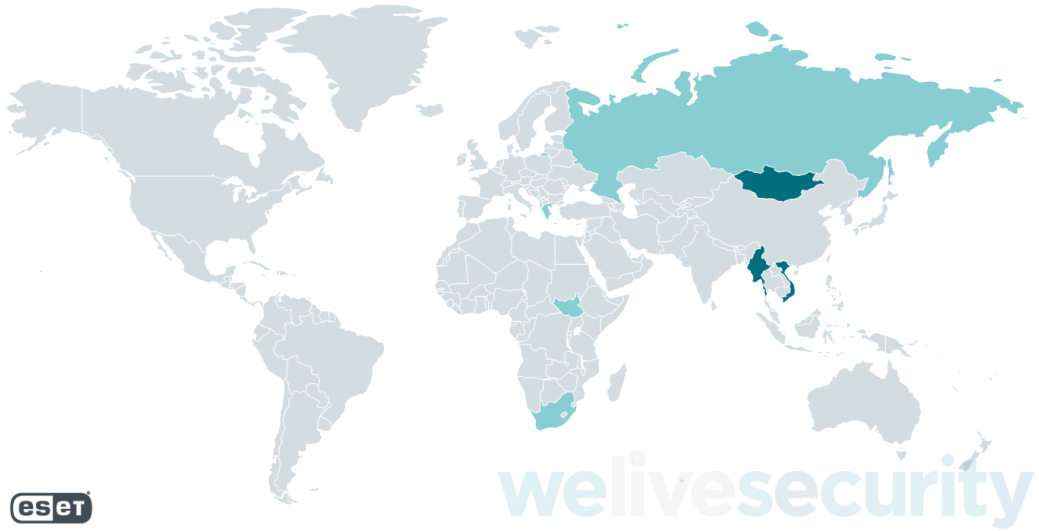

Figure 1. Pays affectés par Mustang Panda dans cette campagne

Pays touchés :

- Mongolie
- Vietnam
- Myanmar

- Grèce
- Russie
- Chypre
- Sud-Soudan
- Afrique du Sud

Verticaux affectés :

- Missions diplomatiques
- Entités de recherche
- Fournisseurs de services Internet (ISP)

Analyse

Sur la base des similitudes de code et des nombreux points communs dans les tactiques, techniques et procédures (TTP), les chercheurs d'ESET attribuent avec un niveau de confiance **élevé** cette campagne à Mustang Panda (également connu sous le nom de TA416, RedDelta, ou PKPLUG). Il s'agit d'un groupe de cyberespionnage qui cible principalement les entités gouvernementales et les ONG. Ses victimes se situent principalement, mais pas exclusivement, en Asie de l'Est et du Sud-Est, tout particulièrement en Mongolie. Le groupe est également connu pour sa campagne visant le Vatican en 2020.

Bien que nous n'ayons pas été en mesure d'identifier les verticaux de toutes les victimes, cette campagne semble avoir les mêmes objectifs de ciblage que les autres campagnes de Mustang Panda. Conformément à la victimologie typique de l'APT, la plupart des victimes sont situées en Asie de l'Est et du Sud-Est, ainsi que dans des pays européens et africains. Selon les données télémétriques d'ESET, la grande majorité des cibles sont situées en Mongolie et au Vietnam, puis au Myanmar, et seulement quelques-unes dans les autres pays touchés.

Les campagnes de Mustang Panda utilisent fréquemment des chargeurs personnalisés pour les logiciels malveillants partagés, notamment Cobalt Strike, Poison Ivy et Korplug (également connu sous le nom de PlugX). Le groupe est également connu pour créer ses propres variantes de Korplug. Par rapport aux autres campagnes utilisant Korplug, chaque étape du processus de déploiement utilise des techniques d'anti-analyse et d'obscurcissement du flux de contrôle.

Cet article de blog présente une analyse détaillée de la variante de Korplug utilisée dans cette campagne, qui n'avait jamais été vue auparavant. Cette activité fait partie de la même campagne récemment couverte par Proofpoint, mais nous fournissons des informations historiques et de ciblage supplémentaires.

Coffre à outil

Mustang Panda est connu pour ses chargeurs personnalisés élaborés et ses variantes de Korplug, et les échantillons utilisés dans cette campagne en sont la parfaite illustration.

Les chaînes de compromission observées dans cette campagne suivent le modèle Korplug typique : un exécutable légitime, valablement signé et vulnérable au détournement de l'ordre de recherche des DLL, une DLL malveillante et un fichier Korplug chiffré sont déployés sur la machine cible. L'exécutable est détourné pour charger le module, qui déchiffre et exécute ensuite le RAT Korplug. Dans certains cas, un téléchargeur est d'abord utilisé pour déployer ces fichiers avec un document de leurre. Ce processus est illustré à la figure 2.

Figure 2. Aperçu du processus de déploiement de la variante Korplug de Hodur

Cette campagne se distingue par l'utilisation intensive de techniques d'obscurcissement du flux de contrôle et d'anti-analyse à chaque étape du processus de déploiement. Les sections suivantes décrivent le comportement de chaque étape et examinent en profondeur les techniques d'évasion de défense utilisées dans chacune d'elles.

Accès initial

Nous n'avons pas été en mesure d'observer le vecteur de déploiement initial, mais notre analyse indique que les attaques de phishing et de watering hole sont des vecteurs probables. Dans les cas où nous avons vu un téléchargeur, les noms de fichiers utilisés suggèrent un document dont le sujet est intéressant pour la cible. En voici des exemples :

- COVID-19 travel restrictions EU reviews list of third countries.exe
- State_aid__Commission_approves_2022-2027_regional_aid_map_for_Greece.exe
- REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL.exe

- Situation at the EU borders with Ukraine.exe

Afin de renforcer l'illusion, ces binaires téléchargent et ouvrent un document portant le même nom mais avec une extension .doc ou .pdf. Le contenu de ces leurres reflète fidèlement le nom du fichier. Comme le montre la figure 3, au moins l'un d'entre eux est un document légitime du Parlement européen accessible au public.

Première page du document leurre pour le téléchargeur REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL.exe . Il s'agit d'un document réel disponible sur le site Web du Conseil européen.

Téléchargeur

Bien que sa complexité ait augmenté au cours de la campagne, le téléchargeur est assez simple. Cette augmentation de la complexité provient de techniques anti-analyse supplémentaires, que nous couvrons plus loin dans cette section.

Il télécharge d'abord quatre fichiers via HTTPS : un document leurre, un exécutable légitime, un module malveillant et un fichier Korplug chiffré. La combinaison de ces trois derniers composants pour exécuter une charge utile par le biais d'un chargement latéral de DLL est parfois appelée *trident* et constitue une technique couramment utilisée par Mustang Panda et par les chargeurs Korplug en général. Les adresses des serveurs et les chemins des fichiers sont codés en dur dans l'exécutable du téléchargeur. Une fois que tout est téléchargé, et que le document leurre est ouvert pour distraire la victime, le téléchargeur utilise la ligne de commande suivante pour lancer l'exécutable légitime :

cmd /c ping 8.8.8.8 -n 70&&"%temp%\<legitimate executable>"

Cette commande ping vérifie la connectivité Internet et introduit un délai (grâce à l'option -n 70) avant d'exécuter l'exécutable légitime téléchargé.

Le téléchargeur utilise de nombreuses techniques anti-analyse, dont beaucoup sont également utilisées dans le chargeur et la charge utile finale. Des techniques d'obscurcissement supplémentaires ont été ajoutées aux nouvelles versions au cours de la campagne sans pour autant modifier leur objectif.

Dans les premières versions du téléchargeur, du code poubelle et des prédicats opaques étaient utilisés pour entraver l'analyse, comme le montre la figure 4, mais le serveur et les noms de fichiers sont clairement visibles en clair.

Figure 4. Obfuscation du flux de contrôle dans les premières versions du téléchargeur

Dans les versions ultérieures, les fichiers sur le serveur sont chiffrés en RC4, en utilisant la représentation de la chaîne de base 10 de la taille du fichier comme clé, puis codés en hexadécimal. Ce processus est illustré dans le snippet Python ci-dessous. Les opérations inverses sont effectuées côté client par le téléchargeur pour récupérer les fichiers en clair. Ceci est probablement fait pour contourner les protections au niveau du réseau.

```
from Crypto.cipher import ARC4
key = "%d" % len(plaintext)
rc4 = ARC4.new(key)
cipher_content = rc4.encrypt(plaintext).hex().upper()
```

Ces versions remplacent l'utilisation de chaînes de texte en clair par des chaînes de pile chiffrées. Elles sont toujours codées en dur dans le fichier, mais l'obscurcissement qui les entoure, et l'utilisation de différentes clés, rendent difficile leur déchiffrement statique de manière automatisée. Cette même technique est fortement utilisée dans les étapes suivantes. Les chaînes de pile chiffrées sont également utilisées pour obscurcir les appels aux fonctions de l'API Windows.

Tout d'abord, le nom de la fonction cible est déchiffré et transmis à une fonction. Cette fonction obtient un pointeur vers le champ InMemoryOrderModuleList du PEB (Process Environment Block). Elle itère ensuite sur les modules chargés, en passant chaque handle à GetProcAddress avec le nom de la fonction jusqu'à ce que la fonction cible soit résolue avec succès. Une partie de ce processus est illustrée à la figure 5.

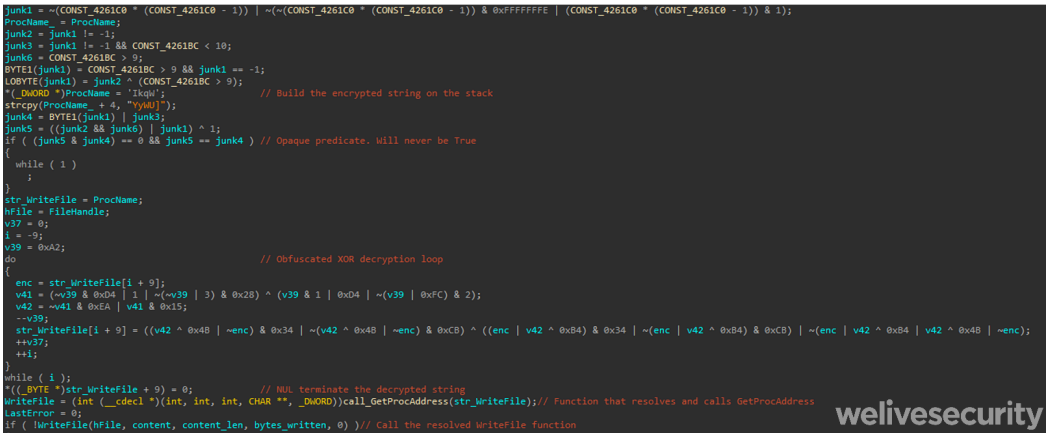

Figure 5. Obfuscation des appels de l'API Windows dans le téléchargeur. La capture d'écran montre un appel à WriteFile, mais le même modèle est utilisé pour toutes les fonctions API

Chargeur

Comme c'est souvent le cas avec Korplug, le chargeur est une DLL qui exploite une vulnérabilité de chargement latéral dans un exécutable légitime et signé. Nous avons observé que de nombreuses applications différentes sont exploitées dans le cadre de cette campagne, par exemple un exécutable SmadAV vulnérable vu précédemment par Qurium dans une campagne attribuée à Mustang Panda qui ciblait le Myanmar.

Le chargeur exporte plusieurs fonctions. La liste exacte varie en fonction de l'application maltraitée, mais dans tous les cas, une seule d'entre elles a une action importante. Dans tous les chargeurs que nous avons observés, il s'agit de la fonction exportée dont l'adresse de chargement est la plus élevée. Toutes les autres exportations, ainsi que le point d'entrée de la

bibliothèque, soit retournent immédiatement, soit exécutent un code inutile. Beaucoup de ces exportations ont des noms qui consistent en des lettres minuscules aléatoires et pointent vers la même adresse, comme le montre le tableau 1.

Tableau 1. Fonctions exportées par un chargeur Hodur. L'exportation createSystemFontsUsingEDL est celle qui charge l'étape finale du logiciel malveillant dans cette version.

Name	Ordinal	Function RVA
CreatePotPlayerExW	1	0x00007894
RunPotPlayer	2	0x000166A5
createSystemFontsUsingEDL	3	0x00016779
gGegcerhwyvxtkrtyawvugo	4	0x00007894
liucigvyworf	5	0x00007639
ojohjinbgdfqtcwxojeusoneslciyxtiyjuieaugadjpd	6	0x000077CA
soeevhiywsypipesxfhgxboleahfwvlqcqp	7	0x00007894
srkeqffanuhiuwahbmatdurggpffhbkcpukyxgxmosn	8	0x00007894
thggvmrv	9	0x00007701

La fonction du chargeur obtient le répertoire dans lequel la DLL s'exécute à l'aide de GetModuleFileNameA et tente d'ouvrir le fichier Korplug chiffré qu'il contient. Ce nom de fichier est codé en dur dans le chargeur. Il lit le contenu du fichier dans un tampon alloué localement et le déchiffre. Le chargeur rend ce tampon exécutable en utilisant VirtualProtect avant de l'appeler à l'offset 0x00.

Les appels aux fonctions de l'API Windows sont obscurcis par une technique différente de celle utilisée dans le téléchargeur. Contrairement au chargeur, qui contient les noms de ses fonctions (comme indiqué dans le tableau 1 ci-dessus), seuls les *hashes* 64 bits des appels de fonctions de l'API Windows sont présents dans le binaire. Pour résoudre ces fonctions, le chargeur parcourt les listes d'exportation de toutes les bibliothèques chargées via la liste InMemoryOrderModuleList du PEB. Le nom de chaque exportation est haché, puis comparé à la valeur attendue. L'algorithme de hachage FNV-1a, récemment remis au goût du jour par la backdoor Sunburst, a déjà été utilisé par Mustang Panda, dans les chargeurs Korplug documentés par XORHEX, pour résoudre GetProcAddress et LoadLibraryA, bien qu'il n'ait pas été identifié par son nom dans cette analyse. Dans cette version, cependant, il est utilisé pour toutes les fonctions de l'API.

Backdoor Korplug

Korplug (également connu sous le nom de PlugX) est un RAT utilisée par plusieurs groupes APT. Bien qu'il soit largement utilisé, ou peut-être à cause de cela, peu de rapports décrivent en détail ses commandes et les données qu'il exfiltre. Ses fonctionnalités ne sont pas constantes d'une variante à l'autre, mais il semble exister un chevauchement important dans la liste des commandes entre la version que nous avons analysée et d'autres sources telles que le *rapport Avira de janvier 2020* et le projet *plugxdecoder* sur GitHub.

Comme mentionné précédemment, la variante utilisée dans cette campagne présente de nombreuses similitudes avec la variante THOR, c'est pourquoi nous l'avons nommée Hodur. Ces similitudes incluent l'utilisation de la clé de registre Software\CLASSES\ms-pu, le même format pour les serveurs C&C dans la configuration et l'utilisation de la classe de fenêtre Static .

Comme prévu pour les charges utiles Korplug, cette étape n'est jamais déchiffrée en mémoire par le chargeur. Seule la version chiffrée est écrite sur le disque dans un fichier avec une extension .dat.

Sauf indication contraire, toutes les chaînes codées en dur dont il est question dans cette section sont stockées sous forme de chaînes de pile chiffrées.

Dans ce module, les fonctions de l'API Windows sont obfusquées par une combinaison des méthodes utilisées dans les étapes précédentes. LoadLibraryA et GetProcAddress sont résolus via la technique de hachage FNV-1a et les chaînes de pile sont déchiffrées et leur sont passées pour obtenir la fonction cible.

Chargement

Une fois déchiffrée, la charge utile est une DLL valide qui exporte une seule fonction. Dans presque tous les échantillons observés de cette campagne, cette fonction s'appelle StartProtect. Cependant, le lancement direct via cette exportation ou son point d'entrée n'exécutera pas la charge utile principale et le processus de chargement est assez complexe.

Comme expliqué dans la section précédente, le fichier est déchiffré en mémoire comme un blob continu par le chargeur et l'exécution commence à l'offset 0x00. L'en-tête du PE contient un shellcode, illustré à la figure 6, qui appelle un offset spécifique correspondant à l'exportation unique du module.

```
HEADER:10000000 ; int __usercall _ImageBase@<eax>(int@<ebp>)
HEADER:10000000 __ImageBase     proc near               ; DATA XREF: HEADER:1000003C↓o
HEADER:10000000                                         ; HEADER:10000110↓o ...
HEADER:10000000                 dec     ebp             ; PE magic number
HEADER:10000001                 pop     edx
HEADER:10000002                 call    $+5             ; Bytes on last page of file
HEADER:10000007                 pop     ebx
HEADER:10000008                 push    edx             ; Size of header in paragraphs
HEADER:10000009                 inc     ebp
HEADER:1000000A                 push    ebp
HEADER:1000000B                 mov     ebp, esp
HEADER:1000000D                 add     ebx, 2C49h
HEADER:10000013                 call    ebx             ; call StartProtect
HEADER:10000015                 leave
HEADER:10000016                 retn
```


Figure 6. Shellcode dans l'en-tête PE qui appelle la fonction exportée

Cette fonction analyse le blob PE en mémoire et le mappe manuellement comme une bibliothèque dans un tampon nouvellement alloué. Cela inclut le mappage des différentes sections, la résolution des importations et, enfin, l'utilisation de DLL_PROCESS_ATTACH pour appeler le point d'entrée de la DLL. Une fois encore, des prédicats opaques et du code poubelle sont utilisés pour obscurcir l'objectif de cette fonction.

Le point d'entrée de la bibliothèque correctement chargée est ensuite appelé avec la valeur non standard de 0x04 pour le paramètre fdwReason (seules les valeurs de 0x00 à 0x03 sont actuellement définies). Cette valeur spéciale est nécessaire pour qu'il exécute sa charge utile principale. Cette simple vérification empêche l'exécution triviale de la RAT directement avec un outil générique comme rundll32.exe.

Le backdoor commence par déchiffrer sa configuration en utilisant la chaîne 123456789 comme clé XOR répétitive. Une fois déchiffré, le bloc de configuration commence par ########. La présentation de la configuration varie légèrement d'un échantillon à l'autre, mais ils contiennent tous au moins les champs suivants :

- Nom du répertoire d'installation. Également utilisé comme nom de la clé de registre créée pour la persistance. Cette valeur correspond approximativement au nom de l'application abusée avec trois lettres aléatoires ajoutées (par exemple, FontEDLZeP ou AdobePhotosGQp).
- Nom du mutex
- Une valeur qui est soit une version, soit une chaîne d'identification
- Liste des serveurs C&C. Chaque entrée comprend l'adresse IP, le numéro de port et un numéro indiquant le protocole à utiliser avec ce C&C.

La backdoor vérifie ensuite le chemin d'accès à partir duquel il s'exécute en utilisant GetModuleFileNameW. S'il correspond à %userprofile%\<installation directory> ou %allusersprofile%\<installation directory>, la fonctionnalité RAT sera exécutée. Sinon, elle passera par le processus d'installation.

Installation

Pour s'installer, le logiciel malveillant crée le répertoire susmentionné sous %allusersprofile%. À l'aide de SetFileAttributesW, il est ensuite marqué comme hidden et system. L'exécutable vulnérable, le module de chargement et les fichiers Korplug chiffrés sont copiés dans le nouveau répertoire.

Ensuite, la persistance est établie. Les échantillons antérieurs y parvenaient en créant une tâche planifiée à exécuter au démarrage via schtasks.exe. Les échantillons plus récents ajoutent une entrée de registre dans Software\Microsoft\Windows\CurrentVersion\Run, en essayant d'abord la ruche HKLM, puis HKCU. Cette entrée porte le même nom que le répertoire d'installation et sa valeur correspond au chemin de l'exécutable nouvellement copié.

Une fois la persistance mise en place, le logiciel malveillant lance l'exécutable depuis son nouvel emplacement et quitte.

RAT

La fonctionnalité de la RAT de la variante de Hodur utilisée dans cette campagne correspond essentiellement à celle des autres variantes de Korplug, avec quelques commandes et caractéristiques supplémentaires. Cependant, comme nous l'avons indiqué précédemment, les analyses détaillées des commandes Korplug sont rares. Nous souhaitons donc fournir une telle analyse dans l'espoir d'aider les futurs analystes.

Dans ce mode, la porte dérobée parcourt la liste des serveurs C&C dans sa configuration jusqu'à ce qu'elle arrive à la fin ou reçoive une commande de désinstallation. Pour chacun de ces serveurs, il traite les commandes jusqu'à ce qu'il reçoive une commande Stop ou qu'il rencontre une erreur.

La poignée de main initiale de Hodur peut se faire par HTTPS ou TCP. Ceci est déterminé par une valeur dans la configuration pour ce serveur C&C particulier. Les communications ultérieures se font toujours via TCP à l'aide d'un protocole personnalisé que nous décrivons dans cette section, ainsi que les commandes qui peuvent être émises. Hodur utilise des sockets de l'API Windows Sockets (Winsock) qui prennent en charge les I/O superposées.

Après la poignée de main initiale, les communications de Hodur impliquent des messages TCP qui se composent d'un en-tête, avec la structure décrite dans le tableau 2, suivi d'un corps de message qui est généralement compressé en utilisant LZNT1 et toujours chiffré avec RC4. Les messages dont le champ d'en-tête Command number a le bit 0x10000000 activé (ceux qui contiennent des contenus de fichiers pour les commandes ReadFile et WriteFile, décrites dans le Tableau 3) ont des corps de message chiffrés mais pas compressés. Tous les corps de message chiffrés utilisent la clé codée en dur sV!e@T#L$PH% à laquelle est ajouté un nonce aléatoire de quatre octets (la valeur située au décalage 0x00 dans l'en-tête).

Tableau 2. Format d'en-tête utilisé pour la communication entre le C&C et le backdoor

Offset	Field	Description
0x00	Nonce	Random nonce appended to the RC4 key.
0x04	Command number	This field indicates the command to run or the command that caused this response to be sent.
0x08	Length of body	Length of the message body. It seems that this field isn’t checked by the client for messages from the C&C server.
0x0C	Command exit status	The return or error value of the command that was run. This field is not checked by the client in messages received from the C&C server.

Les en-têtes des messages du C&C d'Hodur sont transmis en clair, suivis de corps de messages de taille variable (la valeur au décalage 0x08 de l'en-tête). Le format du corps du message varie selon la commande, mais une fois déchifré et décompressé, les valeurs de longueur variable (comme les chaînes de caractères) se trouvent toujours à la fin du corps du message et leur décalage dans le corps est stocké sous forme d'un nombre entier dans le champ du message correspondant.

Comme la version décrite par Avira, Hodur possède deux groupes de commandes – 0x1001 et 0x1002 – chacun ayant son propre gestionnaire. Le serveur C&C peut définir le groupe à écouter en envoyant l'ID correspondant comme numéro de commande lorsqu'un client n'est pas déjà dans l'un des deux modes. Il continuera à écouter le même groupe jusqu'à ce qu'il reçoive la commande Stop, ou qu'une erreur se produise (y compris la réception d'un message dont l'en-tête contient un numéro de commande invalide).

Le premier groupe, 0x1001, contient des commandes permettant de gérer l'exécution de la porte dérobée et d'effectuer une reconnaissance initiale sur un hôte nouvellement compromis. Comme ces commandes ne prennent aucun argument, les messages envoyés par le serveur C&C ne sont constitués que des en-têtes. Le Tableau 3 contient une liste de ces commandes. La commande GetSystemInfo est décrite plus en détail ci-dessous. Notez qu'aucun nom de commande n'est présent dans la RAT ; ils ont été soit repris d'analyses précédentes, soit fournis par nous.

Tableau 3. Commandes du groupe 0x1001

ID	Name	Description	Data in client response
0x1000	Ping	Sent by the client when it starts listening for commands from this group.	Between 0 and 64 random bytes

ID	Name	Description	Data in client response
0x1001	GetSystemInfo	Get information about the system.	See Table 4
0x1002	ListenThread	Start a new thread that listens for group 0x1002 commands.	None
0x1004	ResetConnection	Terminate with WSAECONNRESET.	N/A
0x1005	Uninstall	Delete persistence registry keys, remove itself and created folders.	None
0x1007	Stop	Set registry key System\CurrentControlSet\Control\Network\allow to 1 and exit.	N/A

La commande GetSystemInfo collecte de nombreuses informations sur le système, comme détaillé dans le Tableau 4. Si elle n'existe pas déjà, la clé de registre Software\CLASSES\ms-pu\CLSID est définie avec l'horodatage actuel, en essayant d'abord HKLM puis HKCU. La valeur de cette clé est ensuite envoyée dans la réponse.

Tableau 4. Format du corps de réponse pour la réponse GetSystemInfo

Offset	Value	Offset	Value
0x00	Magic bytes 0x20190301	**0x38**	Suite mask
0x04	Client IP address of the C&C socket	**0x3A**	Product type
0x08	Server IP address of the C&C socket	**0x3C**	0x01 if the process is running as WOW64
0x0C	RAM in KB	**0x40**	System time – year
0x10	CPU clock rate in MHz	**0x42**	System time – month
0x14	Display width in pixels	**0x44**	Timestamp of first run (offset)
0x18	Display height in pixels	**0x46**	Service pack version string (offset)
0x1C	Default locale	**0x48**	*Unknown*
0x20	Current tick count	**0x4A**	Username (offset)
0x24	OS major version	**0x4C**	Computer name (offset)
0x28	OS minor version	**0x4E**	Mutex name (offset)
0x2C	OS build number	**0x50**	*Unknown*
0x30	OS platform ID	**0x52**	List of machine IP addresses (offset)
0x34	Service pack major version	**0x54**	Always two 0x00 bytes
0x36	Service pack minor version		

Le groupe 0x1002 contient des commandes qui fournissent la fonctionnalité de la RAT, comme détaillé dans le Tableau 5. Certaines d'entre elles prennent des paramètres fournis dans le corps du message de la commande. La commande FindFiles est décrite plus en détail ci-dessous. Encore une fois, notez qu'aucun nom de commande n'est présent dans la RAT ; ils ont été soit repris d'analyses précédentes, soit fournis par nous.

Tableau 5. Commandes du groupe 0x1002

ID	Name	Description	Data in C&C request	Data in client response
0x1002	Ping	Sent by the client when it starts listening for commands from this group.	N/A	None

ID	Name	Description	Data in C&C request	Data in client response
0x3000	ListDrives	List all mapped drives (A: to Z:) and their properties. All 26 entries are sent back in one message body. Drives that aren't present have all fields set to 0x00.	None	• Drive type • Total size • Space available to user • Free space • Volume name (offset) • File system name (offset)
0x3001	ListDirectory	List the contents of the specified directory. The client sends one response message per entry.	Directory path	• Is a directory? • File attributes • File size • Creation time • Last write time • Filename (offset) • 8.3 filename (offset)
0x3002	#rowspan#	Sent by the client when it has finished executing the ListDirectory command.	N/A	None
0x3004	ReadFile	Read a file in chunks of 0x4000 bytes.	• Creation time • Last access time • Last write time • Has offset • Offset in file • File size • File path	
0x10003005	#rowspan#	Chunk of read file data.	N/A	Read data
0x10003006	#rowspan#	Sent by the client when it has finished executing the ReadFile command.	N/A	None
0x3007	WriteFile	Write to a file and restore previous timestamp. Creates parent directories if they don't exist.	• Creation time • Last access time • Last write time • Has offset • Offset in file • File path (offset)	None
0x10003008	#rowspan#	Sent by the server with data to write to the file.	Data to write	N/A
0x10003009	#rowspan#	Sent by the server when the WriteFile operation is complete.	None	N/A
0x300A	CreateDirectory	Create a directory.	Directory path	None
0x300B	CanReadFile	Try to open a file with read permissions.	File path	None

ID	Name	Description	Data in C&C request	Data in client response
0x300C	DesktopExecute	Execute a command on a hidden desktop.	Command line to execute	PROCESS_INFORMATION structure for the created process.
0x300D	FileOperation	Perform a file operation using SHFileOperation.	• wFunc • fFlags • pFrom (offset) • pTo (offset)	None
0x300E	GetEnvValue	Get the value of an environment variable.	Environment variable	Environment variable value.
0x300F	CreateProgramDataDir	Creates the directory %SYSTEM%\ProgramData, optionally with a subdirectory.	Subdirectory relative path (optional)	None
0x3102	FindFiles	Recursively search a directory for files matching a given pattern.	• Starting directory • Search pattern	See response body format in Table 6.
0x7002	RemoteShell	Start an interactive remote cmd.exe session.	None	None
0x7003	#rowspan#	Result of the last command run.	N/A	Command output

FindFiles command

À partir du répertoire fourni, cette commande recherche les fichiers dont le nom correspond au motif donné. Ce modèle prend en charge les mêmes caractères génériques que l'API Windows FindFirstFile. Pour chaque fichier correspondant, le client envoie un message de réponse dont le corps a le format décrit dans le Tableau 6.

Tableau 6. Format du corps de la réponse pour la commande FindFiles

Offset	Value	Offset	Value
0x00	File attributes	**0x24**	Folder path (offset)
0x04	File size in bytes	**0x26**	Filename (offset)
0x0C	Creation time	**0x28**	8.3 filename (offset)
0x1C	Last write time		

Un message de réponse avec un corps sans contenu est envoyé une fois la recherche terminée.

Conclusion

Les leurres utilisés dans cette campagne montrent une fois de plus la rapidité avec laquelle Mustang Panda est capable de réagir aux événements mondiaux. Par exemple, un règlement de l'UE sur le COVID-19 a été utilisé comme leurre deux semaines seulement après sa publication, et des documents sur la guerre en Ukraine ont commencé à être utilisés dans les jours qui ont suivi le début du lancement de l'invasion. Ce groupe fait également preuve d'une capacité à améliorer ses outils de manière itérative, y compris son utilisation caractéristique de téléchargeurs tridentés pour déployer Korplug.

Pour toute question concernant nos recherches publiées sur WeLiveSecurity, veuillez nous contacter à threatintel@eset.com. ESET Recherche propose désormais des rapports privés de renseignements sur les groupes APT et des flux de données. Pour toute demande de renseignements sur ce service, visitez la page ESET Threat Intelligence.

Indicateurs de compromission (IoCs)

SHA-1	Filename	ESET detection name	Descr
69AB6B9906F8DCE03B43BEBB7A07189A69DC507B	coreclr.dll	Win32/Agent.ADMW	Korpl
10AE4784D0FFBC9CD5FD85B150830AEA3334A1DE	N/A	Win32/Korplug.TC	Decry (dum memo
69AB6B9906F8DCE03B43BEBB7A07189A69DC507B	coreclr.dll	Win32/Agent.ADMW	Korpl
4EBFC035179CD72D323F0AB357537C094A276E6D	PowerDVD18.exe	Win32/Delf.UTN	Korpl
FDBB16B8BA7724659BAB5B2E1385CFD476F10607	N/A	Win32/Korplug.TB	Decry (dum memo
7E059258CF963B95BDE479D1C374A4C300624986	N/A	Win32/Korplug.TC	Decry (dum memo
7992729769760ECAB37F2AA32DE4E61E77828547	SHELLSEL.ocx	Win32/Agent.ADMW	Korpl
F05E89D031D051159778A79D81685B62AFF4E3F9	SymHp.exe	Win32/Delf.UTN	Korpl
AB01E099872A094DC779890171A11764DE8B4360	BoomerangLib.dll	Win32/Korplug.TH	Korpl
CDB15B1ED97985D944F883AF05483990E02A49F7	PotPlayer.dll	Win32/Agent.ADYO	Korpl
908F55D21CCC2E14D4FF65A7A38E26593A0D9A70	SmadHook32.dll	Win32/Agent.ADMW	Korpl
477A1CE31353E8C26A8F4E02C1D378295B302C9E	N/A	Win32/Agent.ADMW	Korpl
52288C2CDB5926ECC970B2166943C9D4453F5E92	SmadHook32c.dll	Win32/Agent.ADMW	Korpl
CBD875EE456C84F9E87EC392750D69A75FB6B23A	SHELLSEL.ocx	Win32/Agent.ADMW	Korpl
2CF4BAFE062D38FAF4772A7D1067B80339C2CE82	Adobe_Caps.dll	Win32/Agent.ADMW	Korpl
97C92ADD7145CF9386ABD5527A8BCD6FABF9A148	DocConvDll.dll	Win32/Agent.ADYO	Korpl
39863CECA1B0F54F5C063B3015B776CDB05971F3	N/A	Win32/Korplug.TD	Decry (dum memo
0D5348B5C9A66C743615E819AEF152FB5B0DAB97	FontEDL.exe	clean	Vulne legiti File C execu
C8F5825499315EAF4B5046FF79AC9553E71AD1C0	Silverlight.Configuration.exe	clean	Vulne legiti Micro Silve Confi Utilit
D4FFE4A4F2BD2C19FF26139800C18339087E39CD	PowerDVDLP.exe	clean	Vulne legiti Powe execu
65898ACA030DCEFDA7C970D3A311E8EA7FFC844A	Symantec.exe	clean	Vulne legiti

SHA-1	Filename	ESET detection name	Desc
			Syma AntiV execu
7DDB61872830F4A0E6BF96FAF665337D01F164FC	Adobe Stock Photos CS3.exe	clean	Vulne legiti Stock execu
C13D0D669365DFAFF9C472E615A611E058EBF596	COVID-19 travel restrictions EU reviews list of third countries.exe	Win32/Agent_AGen.NJ	Down
062473912692F7A3FAB8485101D4FCF6D704ED23	REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL.exe	Win32/TrojanDownloader.Agent.GDL	Down
2B5D6BB5188895DA4928DD310C7C897F51AAA050	log.dll	Win32/Agent.ACYW	Korpl
511DA645A7282FB84FF18C33398E67D7661FD663	2.exe	Win32/Agent.ADPL	Korpl
59002E1A58065D7248CD9D7DD62C3F865813EEE6	log.dll	Win32/Agent.ADXE	Korpl
F67C553678B7857D1BBC488040EA90E6C52946B3	KINGSTON.exe	Win32/Agent.ADXZ	Korpl
58B6B5FD3F2BFD182622F547A93222A4AFDF4E76	PotPlayer.exe	clean	Vulne legiti execu

Réseaux

Domain	IP	First seen	Notes
	103.56.53[.]120	2021-06-15	Korplug C&C
	154.204.27[.]181	2020-10-05	Korplug C&C.
	43.254.218[.]42	2021-02-09	Download server.
	45.131.179[.]179	2020-10-05	Korplug C&C.
	176.113.69[.]91	2021-04-19	Korplug C&C.
upespr[.]com	45.154.14[.]235	2022-01-17	Download server.
urmsec[.]com	156.226.173[.]23	2022-02-23	Download server.
	101.36.125[.]203	2021-06-01	Korplug C&C.
	185.207.153[.]208	2022-02-03	Download server.
	154.204.27[.]130	2021-12-14	Korplug C&C.
	92.118.188[.]78	2022-01-27	Korplug C&C.
zyber-i[.]com	107.178.71[.]211	2022-03-01	Download server.
locvnpt[.]com	103.79.120[.]66	2021-05-21	Download server. This domain was previously used in a 2020 campaign documented by Recorded Future.

Techniques MITRE ATT&CK

Ce tableau a été conçu en utilisant la *version 10* de MITRE ATT&CK.

Tactic	ID	Name	Description
Resource Development	T1583.001	Acquire Infrastructure: Domains	Mustang Panda has registered domains for use as download servers.
	T1583.003	Acquire Infrastructure: Virtual Private Server	Some download servers used by Mustang Panda appear to be on shared hosting.
	T1583.004	Acquire Infrastructure: Server	Mustang Panda uses servers that appear to be exclusive to the group.
	T1587.001	Develop Capabilities: Malware	Mustang Panda has developed custom loader and Korplug versions.
	T1588.006	Obtain Capabilities: Vulnerabilities	Multiple DLL hijacking vulnerabilities are used in the deployment process.
	T1608.001	Stage Capabilities: Upload Malware	Malicious payloads are hosted on the download servers.
Execution	T1059.003	Command and Scripting Interpreter: Windows Command Shell	Windows command shell is used to execute commands sent by the C&C server.
	T1106	Native API	Mustang Panda uses CreateProcess and ShellExecute for execution.
	T1129	Shared Modules	Mustang Panda uses LoadLibrary to load additional DLLs at runtime. The loader and RAT are DLLs.
	T1204.002	User Execution: Malicious File	Mustang Panda relies on the user executing the initial downloader.
	T1574.002	Hijack Execution Flow: DLL Side-Loading	The downloader obtains and launches a vulnerable application so it loads and executes the malicious DLL that contains the second stage.
Persistence	T1547.001	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder	Korplug can persist via registry Run keys.
	T1053.005	Scheduled Task/Job: Scheduled Task	Korplug can persist by creating a scheduled task that runs on startup.
Defense Evasion	T1140	Deobfuscate/Decode Files or Information	The Korplug file is encrypted and only decrypted at runtime, and its configuration data is encrypted with XOR.
	T1564.001	Hide Artifacts: Hidden Files and Directories	Directories created during the installation process are set as hidden system directories.
	T1564.003	Hide Artifacts: Hidden Window	Korplug can run commands on a hidden desktop. Multiple hidden windows are used during the deployment process.
	T1070	Indicator Removal on Host	Korplug's uninstall command deletes registry keys that store data and provide persistence.
	T1070.004	Indicator Removal on Host: File Deletion	Korplug can remove itself and all created directories.
	T1070.006	Indicator Removal on Host: Timestomp	When writing to a file, Korplug sets the file's timestamps to their previous values.

Tactic	ID	Name	Description
	T1036.004	Masquerading: Masquerade Task or Service	Scheduled tasks created for persistence use legitimate-looking names.
	T1036.005	Masquerading: Match Legitimate Name or Location	File and directory names match expected values for the legitimate app that is abused by the loader.
	T1112	Modify Registry	Korplug can create, modify, and remove registry keys.
	T1027	Obfuscated Files or Information	Some downloaded files are encrypted and stored as hexadecimal strings.
	T1027.005	Obfuscated Files or Information: Indicator Removal from Tools	Imports are hidden by dynamic resolution of API function names.
	T1055.001	Process Injection: Dynamic-link Library Injection	Some versions of the Korplug loader inject the Korplug DLL into a newly launched process.
	T1620	Reflective Code Loading	Korplug parses and loads itself into memory.
Discovery	T1083	File and Directory Discovery	Korplug can list files and directories along with their attributes and content.
	T1082	System Information Discovery	Korplug collects extensive information about the system including uptime, Windows version, CPU clock rate, amount of RAM and display resolution.
	T1614	System Location Discovery	Korplug retrieves the system locale using GetSystemDefaultLCID.
	T1016	System Network Configuration Discovery	Korplug collects the system hostname and IP addresses.
	T1016.001	System Network Configuration Discovery: Internet Connection Discovery	The downloader pings Google's DNS server to check internet connectivity.
	T1033	System Owner/User Discovery	Korplug obtains the current user's username.
	T1124	System Time Discovery	Korplug uses GetSystemTime to retrieve the current system time.
Collection	T1005	Data from Local System	Korplug collects extensive data about the system it's running on.
	T1025	Data from Removable Media	Korplug can collect metadata and content from all mapped drives.
	T1039	Data from Network Shared Drive	Korplug can collect metadata and content from all mapped drives.
Command and Control	T1071.001	Application Layer Protocol: Web Protocols	Korplug can make the initial handshake over HTTPS.
	T1095	Non-Application Layer Protocol	C&C communication is done over a custom TCP-based protocol.
	T1573.001	Encrypted Channel: Symmetric Cryptography	C&C communication is encrypted using RC4.
	T1008	Fallback Channels	The Korplug configuration contains fallback C&C servers.

Tactic	ID	Name	Description
	T1105	Ingress Tool Transfer	Korplug can download additional files from the C&C server.
	T1571	Non-Standard Port	When Hodur performs its initial handshake over HTTPS, it uses the same port (specified in the configuration) as for the rest of the communication.
	T1132.001	Data Encoding: Standard Encoding	Korplug compresses transferred data using LZNT1.
Exfiltration	T1041	Exfiltration Over C2 Channel	Data exfiltration is done via the same custom protocol used to send and receive commands.
