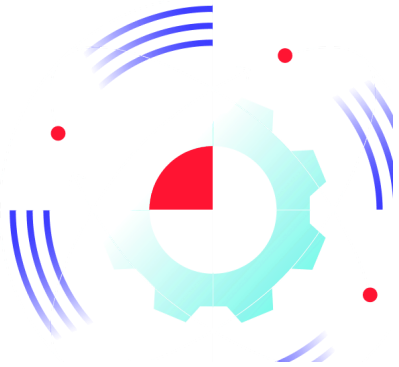
Utilisation de faux profils Steam : Vidar Stealer prend les commandes - Gatewatcher

Archived: 2026-04-05 22:45:43 UTC

Depuis quelques années, l'utilisation d'infostealer est en forte croissance. Ces outils sont parfois utilisés par les acteurs malveillants threats actors pour réaliser des accès initiaux sur des systèmes ou pour la collecte de données qui seront ensuite revendues sur les forums. Les analystes de chez Gatewatcher ont eu l'occasion d'analyser bon nombre de menaces relatives au vol de données.



Etude du logiciel

C'est dans ce contexte d'analyse, et pour faire suite à un article récent portant sur une technique d'évasion utilisant des archives zip que nous avons étudié ce logiciel malveillant.

Lors de nos analyses quotidiennes, une archive contenant une vidéo de présentation d'un jeu vidéo ludique et un fichier portable exécutable modifié pour tromper la vigilance de la victime a été observée. Pour cette modification, l'attaquant a utilisé deux éléments :

- L'utilisation de l'extension « *.scr* » afin de dissimuler l'extension habituelle « *.exe* » utilisée par ce malware.
- La modification de l'icône de l'exécutable. Dans notre cas, l'attaquant a remplacé l'icône de l'exécutable par celle d'une image, ajoutant également le nom « passeports scan […] » dans le titre, pouvant laisser penser à l'utilisateur que ce fichier est un scan de passeport.

Dans un premier temps, nous avons remarqué que le fichier exécutable était assez lourd. En effet, l'archive contenant la vidéo et l'exécutable faisait et occupant environ 70 Mo, il était difficilement concevable que cette archive contienne un fichier PE de 680Mo.

Lors d'une précédente note, nous avions expliqué comment certains malwares augmentent artificiellement leur taille en utilisant du *padding*. Nous avons donc utilisé la même méthode que celle décrite précédemment pour enlever le *padding* et obtenir un fichier plus petit.

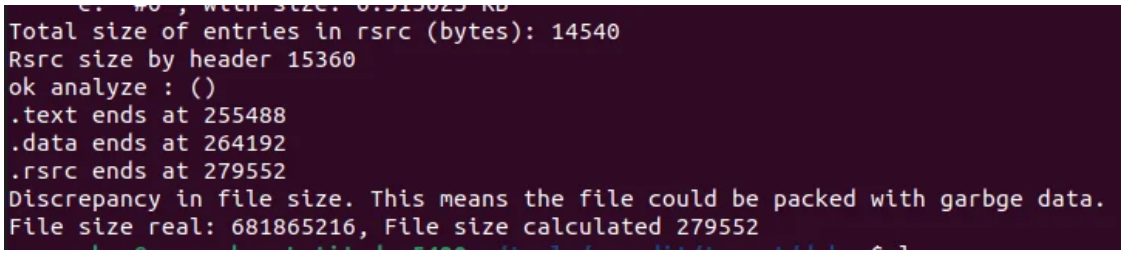


Image 1 : résultat de l'outil permettant de retirer le padding

Suite aux manipulations effectuées, le fichier est réduit à 279Ko, ce qui le rend davantage exploitable pour le reste de nos analyses. Une fois l'exécutable récupéré, il est envoyé en sandbox pour analyse. Le malware est finalement identifié comme une souche de Vidar stealer, un stealer basé sur arkei stealer et permettant le vol et l'exfiltration de données sensibles telles que des données bancaires, mots de passe et autres éléments stockés dans des navigateurs.

Redirection vers le serveur de commande

Lors de l'analyse, une particularité du malware a attiré notre attention. En effet, le premier contact du serveur n'est pas fait sur un serveur de contrôle mais sur deux adresses correspondant à un profil Steam et un Canal T*elegram*.

En examinant de plus près le nom de joueur du profil *Steam* contacté, nous voyons que ce dernier utilise un nom aléatoire et une *URL* contenant une adresse *IP*, le tout finissant par un *pipe*. Lorsque nous nous penchons d'un peu plus près sur l'historique des noms utilisés par ce compte, nous remarquons que ces derniers utilisent un schéma récurrent. La seule partie qui diffère correspond à l'adresse *IP*:

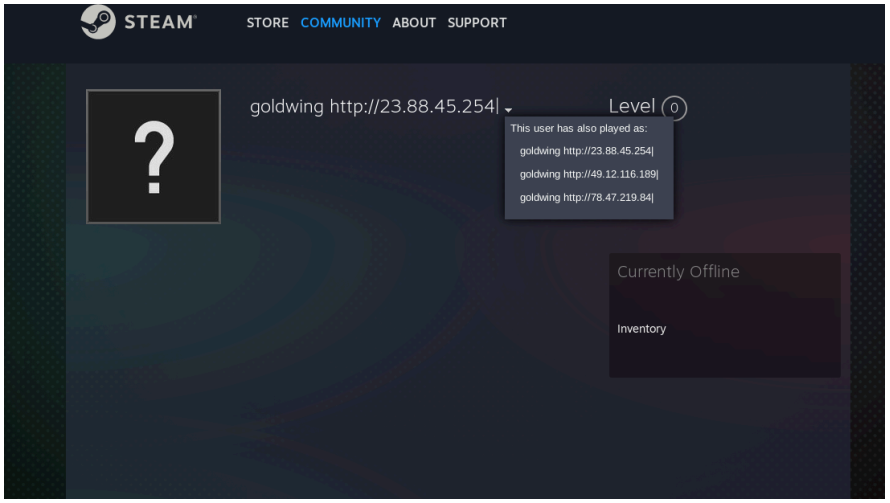


Image 2 : capture du compte steam de redirection

Cette technique de changement de nom d'utilisateur pseudo sur *Steam* permet une modification aisée du serveur de commande sans avoir à modifier le code malveillant, ce qui représente un gain de temps et facilite la

propagation des implants malveillants. En effet, lorsqu'une infrastructure est identifiée comme un serveur de contrôle pour Vidar, il suffit simplement de supprimer le serveur détecté, d'en créer un nouveau et de changer le nom d'un compte *Steam* pour modifier la redirection vers le nouveau serveur de commande. Ici, nous pouvons observer que seules les adresses *IP* ont changé. Cette information couplée au nom « *goldwing* » pourrait nous indiquer qu'une campagne est en cours et que toutes les machines possédant ces adresses *IP* font ou ont fait partie d'une campagne utilisant l'implant analysé.

Ce n'est pas la première fois que nous remarquons l'utilisation de profils *Steam* (ou de fausses pages de profils *Steam*) afin de transmettre l'adresse du serveur de commande au malware. Dans les 6 derniers mois, nous avons recensé pas moins de 350 pages de ce type sur *Steam*. De plus, la majoritéde ces profils utilisent le même schéma : un nom suivi d'une adresse IP. Voici d'autres exemples de redirection qui ne sont pas liés à notre implant mais qui sont relatif à vidar stealer :

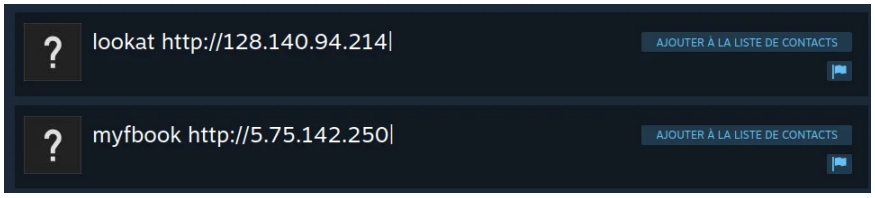


Image 3 : capture d'autres compte steam de redirection

Passons maintenant au deuxième cas, celui-ci plus habituel, un canal Telegram hébergeant l'URL de redirection qui sera utilisé dans la suite de l'attaque. Ce Telegram se présente de la manière suivante :

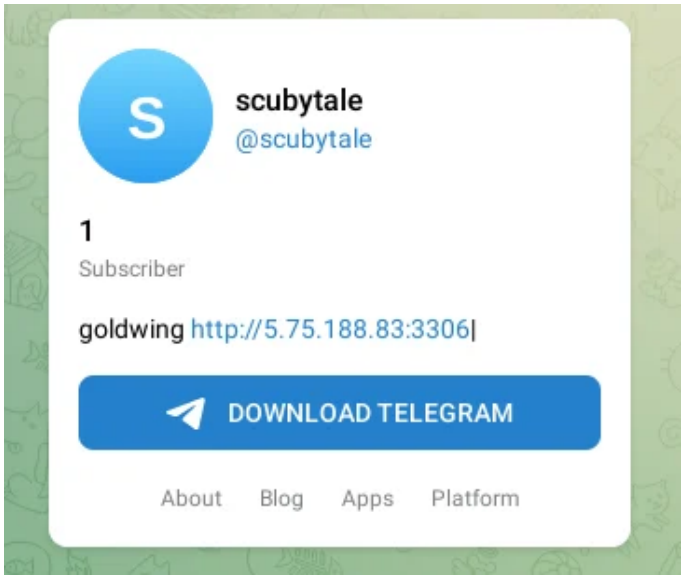


Image 4 : capture du compte telegram de redirection

Il s'agit de la même technique que celle utilisée pour le compte steam utilisant, de surcroît, le même nom, "*goldwing*". Malheureusement, les canaux *Telegram* ne nous offrent pas les mêmes possibilités en termes d'historique, et représentent même une complexité supplémentaire pour l'attaquant. En effet, lorsque pour *Steam* il suffit d'un email et d'un mot de passe, T*elegram* lui nécessite un numéro de téléphone ce qui peut être plus contraignant pour les attaquants. Ce canal aura été utilisé pendant moins d'un mois par notre implant.

Nous pouvons, dès lors, nous interroger sur la raison d'utiliser *Steam* et *Telegram* pour indiquer à l'implant son serveur de commande. En dehors de l'agilité que procure ce mode opératoire, il permet également de changer de serveur de commande lorsque celui-ci est identifié comme malveillant. Cette pratique est utile lorsque l'attaquant cherche à échapper aux systèmes de détection.

L'avantage supplémentaire de cette méthode est qu'elle évite de renseigner un serveur de commande directement dans le code du malware.

Analyse des échanges et des éléments laissés par le malware

Passons maintenant au fonctionnement de l'implant. Dans le cas de *Vidar Stealer*, le serveur de commande va être utilisé pour réaliser plusieurs actions :

- Indiquer à l'implant quelles capacités déployer
- Indiquer aux malwares quelles informations récupérer sur le poste infecté
- Récupérer les données volées

Dans un premier temps, l'implant va contacter le serveur de commande afin de récupérer son paramétrage et les DLL nécessaires à son exécution :

```
Data Ascii: 3c1,1,1,1,1,820c53c3005da334cce09ca619710ee2,1,1,0,1,0,none;,00
```

Image 5 : commande de paramétrage du C2

Pour mieux comprendre les actions à effectuer par l'implant demandé par le serveur, nous nous sommes appuyés sur l'article « Vidar Stealer H&M Campaign – deep dive analysis of a Vidar Stealer »[1]. En suivant le découpage expliqué par l'auteur de l'article, voici ce que nous avons pu extraire des informations qui seront récupérées et envoyées au serveur :

- 3C correspond aux données de référence de l'implant
- 1,1,1,1,1 cette série de 1 correspond à la configuration de l'extraction (dans notre cas, les mots de passe locaux, cookies, portefeuilles crypto, historique du navigateur et les données Telegrams seront extraits)
- 820c53c3005da334cce09ca619710ee2 correspond au token d'exfiltration
- 1,1,0, cette deuxième série correspond à l'extraction d'autres données (dans notre cas les données discord et Steam seront extraites mais aucun Screenshot ne sera pris)
- 1,0 cette troisième série indiqueà l'implant qu'il pourra utiliser un grabber et que pour toutes les extractions signifiées précédemment, il n'y aura pas de limite de taille de fichier.
- None signifie qu'aucun profil en particulier ne sera ciblé
- ; ,00 signifie qu'aucun répertoire spécifique n'est utilisé par le stealer afin de collecter des données et qu'aucune limite de taille de données n'est à considérer

Une fois le paramétrage reçu, l'implant procède au téléchargement des *DLL* nécessaires à son exécution. Les *DLL* ou *Dynamic Link Library* sont des bibliothèques qui contiennent du code pouvant être appelé par des programmes,

dans notre cas, par *vidar stealer*. Voici les *DLL* qui seront téléchargées dans le répertoire d'exécution du *malware* :

- vcruntime140.dll
- softokn3.dll
- nss3.dll
- msvcp140.dll
- mozglue.dll
- freebl3.dll

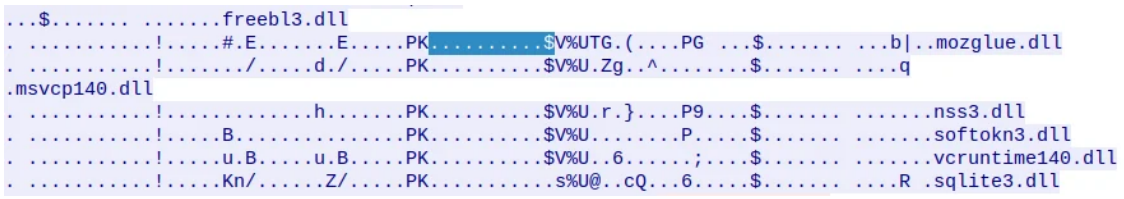


Image 6 : capture du tcp stream du telechargement des dll

Bien que la plupart de ces *DLL* sont présentes dans les systèmes Windows, le téléchargement de ces dernières permet d'échapper aux détections et de minimiser autant que possible, de laisser des traces sur le système infecté.

Pour bien comprendre pourquoi il est intéressant pour l'attaquant de télécharger des *DLL*, il est nécessaire de comprendre l'ordre de recherche de *ces fichiers par les programmes*. Lorsqu'un programme a besoin de *DLL* il va d'abord chercher si elles ne sont pas déjà chargées en mémoire, puis, rechercher dans différents répertoires courants accessible par l'utilisateur (en commençant par le répertoire d'exécution du programme), pour enfin, rechercher dans des répertoires systèmes où sont stockés les *DLL*, ce qui nécessite, généralement des droits élevés. La technique de téléchargement des *DLL* dans le répertoire d'exécution du programme malveillant permet donc deux choses :

- S'assurer que l''environnement du malware est correctement configuré pour pouvoir opérer
- Ne pas se soucier des droits de l'utilisateur puisque les DLL sont dans le répertoire courant d'utilisation et de ne pas lever d'alerte relative à l'accession ou la tentative d'accès par un programme inconnu à des répertoire nécessitant des droits élevés.

Si vous souhaitez plus de détails sur les techniques d'attaques en lien avec DLL et les contre-mesures associés, vous pouvez consulter notre cyber threat semester report.

Grâce à ces *DLL* légitimes, le *stealer* est enfin prêt à opérer et créer des bases de données *SQLite*. Dans notre cas, ces bases de données contiennent 4 types de data :

- Des données bancaires
- Des données relatives à l'historique des navigateurs
- Les données des cookies du navigateur
- Les données relatives aux mots de passe présents sur le système

Voici un exemple de structure de base utilisé pour sauvegarder des données bancaires:

Image 7 : exemple de structure de table des bases de données du stealer

Ces données sont ensuite envoyées, en clair, sous la forme d'une archive zip encodé en base64, au serveur de commande. Cette archive contient les informations de la machine cible et les données volées.

Afin de comprendre le fonctionnement de cet implant nous avons dû faire appel à plusieurs outils. Lors de nos premières tentatives de compréhension de son fonctionnement nous avons lancé le *stealer* dans nos s*andbox* puis dans des *sandboxs* du marché. Alors que dans nos *sandbox*, nous obtenions la chaine de fonctionnement complète, les autres environnements de tests s'arrêtaient brusquement après la création des bases de données SQLite. En analysant cet arrêt brusque, les codes d'erreurs associés, le code du malware et la littérature autour de cet exécutable, nous avons pu identifier que certains mécanismes d'évasion des systèmes d'analyses existaient dans le code. Dans notre cas, lorsqu'une analyse dynamique avec un debugger était identifié, le malware s'arrêtait en créant l'erreur c000005 qui correspond à l' »Access violation error » qui conduit, dans notre cas, à une corruption de la mémoire vive. Cette fonctionnalité permet entre autres d'éviter la récupération pour analyse de la RAM lors de l'exécution du malware.

Conclusion

Vidar stealer est un des *stealer* les plus utilisés dans le monde tant pour son efficacité que pour sa facilité de prise en main. Au travers de cet exemple, nous avons pu montrer comment les attaquants utilisent l'ingénierie sociale ainsi que différentes techniques de dissimulation et d'évasion de détection pour réaliser leurs exactions. Au travers de l'étude du comportement de vidar, nous avons pu identifier un certain nombre d'éléments nous permettant de mener des actions de *Threat Hunting* comme :

- La recherche de compte suivant le pattern décrit précédemment (un nom suivi d'une *URL* se finissant par un *pipe*) dans la communauté Steam

- L'apparition de canal T*elegram* contenant dans la description un même pattern. Que celui des comptes Steam

Ce threat Hunting est très important car il nous permet de construire une base solide de connaissance nécessaire à l'amélioration des règles de détection. Cette base est également importante sur un autre aspect, elle permet de donner du contexte en cas de réponse à incident afin de comprendre au mieux les causes et les potentiels dommages qui peuvent être engendrés suite à une attaque.

En outre, l'intégration du *Threat Hunting* combiné aux analyses d'indices de compromission intégrés quotidiennement permet à GATEWATCHER d'alimenter une base de connaissance actionnable dans tous ses produits de détections.

[1] « Vidar Stealer H&M Campaign – deep dive analysis of a Vidar Stealer », Threat Analyst & IR team leader – Malware Analysis – Blue Team

***Auteurs** : Purple Team GATEWATCHER et 0xSeeker*

Annexes

TTPs évoqués dans cet article

Mitre Att&ck Matrix

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Exfiltration	Command and Control	Network Effects	Remote Service Effects	Impact
Valid Accounts	1 Windows Management Instrumentation	Path Interception	1 Process Injection	1 1 Disable or Modify Tools	1 OS Credential Dumping	2 System Time Discovery	Remote Services	1 Archive Collected Data	Exfiltration Over Other Network Medium	2 1 Encrypted Channel	Eavesdrop on Insecure Network Communication	Remotely Track Device Without Authorization	Modify System Partition
Default Accounts	2 Command and Scripting Interpreter	Boot or Logon Initialization Scripts	Boot or Logon Initialization Scripts	1 Process Injection	1 Credentials in Registry	4 1 Security Software Discovery	Remote Desktop Protocol	3 Data from Local System	Exfiltration Over Bluetooth	1 1 Non-Standard Port	Exploit SS7 to Redirect Phone Calls/SMS	Remotely Wipe Data Without Authorization	Device Lockout
Domain Accounts	2 Native API	Logon Script (Windows)	Logon Script (Windows)	1 Deobfuscate/Decode Files or Information	Security Account Manager	2 Process Discovery	SMB/Windows Admin Shares	Data from Network Shared Drive	Automated Exfiltration	3 Ingress Tool Transfer	Exploit SS7 to Track Device Location	Obtain Device Cloud Backups	Delete Device Data
Local Accounts	At (Windows)	Logon Script (Mac)	Logon Script (Mac)	3 Obfuscated Files or Information	NTDS	1 Account Discovery	Distributed Component Object Model	Input Capture	Scheduled Transfer	4 Non-Application Layer Protocol	SIM Card Swap		Carrier Billing Fraud
Cloud Accounts	Cron	Network Logon Script	Network Logon Script	2 2 Software Packing	LSA Secrets	1 System Owner/User Discovery	SSH	Keylogging	Data Transfer Size Limits	1 1 3 Application Layer Protocol	Manipulate Device Communication		Manipulate App Store Rankings or Ratings
Replication Through Removable Media	Launchd	Rc.common	Rc.common	Steganography	Cached Domain Credentials	3 File and Directory Discovery	VNC	GUI Input Capture	Exfiltration Over C2 Channel	Multiband Communication	Jamming or Denial of Service		Abuse Accessibility Features
External Remote Services	Scheduled Task	Startup Items	Startup Items	Compile After Delivery	DCSync	4 4 System Information Discovery	Windows Remote Management	Web Portal Capture	Exfiltration Over Alternative Protocol	Commonly Used Port	Rogue Wi-Fi Access Points		Data Encrypted for Impact

Détection suricata associée

Lors de nos recherches sur les moyens de détection de vidar, nous avons pu observer qu'un certain nombre de règles suricata existaient déjà et qu'elles permettaient de détecter les signatures réseaux de Vidar Stealer. Les capacités de vidar étant en perpétuelle évolution, nous avons décidé d'ajouter à notre arsenal de détection une règle qui nous permettra de détecter l'extraction d'un zip encodé en base64, la voici :

alert http any any -> any any (msg: »PURPLE RULES detection of vidar stealer extraction file via Base64 zip »;flow:established,to_server;http.method; content: »POST »; http.request_body;content: »Content-Disposition:|20|form-data|3B 20|name=|22|hwid|22| »;content: »|0d 0a 0d

0a|UEs »;content: »L2luZm9ybWF0aW9uLnR4d »; distance:0;sid:1000101;rev:1; metadata: provider Gatewatcher, signature_severity Critical, risk 95;)

IoCs

Type	iocs	Description
Hash	b31d8ed2ec59462f275ed8c3b158c51f	MD5 de l'implant décrit dans cet article
Hash	48f0088e73b22e506efd6fc229b95f5adb87abfc	SHA1 de l'implant décrit dans cet article
Hash	bcb9ad2db4cbbb3dbd270895b570c587a1b59ef15edc388938a40700c6efbeee	SHA256 de l'implant décrit dans cet article
Url	https://t.me/scubytale	Canal telegram renseignant le serveur de commande pour le malware décris dans cet article
Url	https://*steam*community.com/profiles/76561199564671869	Compte Steam renseignant le serveur de commande pour le malware décris dans cet article
Url	http://23.88.45.254	Url du C2 Vidar de la campagne goldwing
IP	23[.]88[.]45[.]254	@ip du C2 vidar de la campagne goldwing
Url	http://49.12.116.189	Url du C2 Vidar de la campagne goldwing

IP	49[.]12[.]116[.]189	@ip du C2 vidar de la campagne goldwing
Url	http://78.47.219.84	Url du C2 Vidar de la campagne goldwing
IP	78[.]47[.]219[.]84	@ip du C2 vidar de la campagne goldwing
Url	http://5.75.188.83:3306	Url du C2 Vidar de la campagne goldwing utilisé par cet implant dans ce rapport
IP	5[.]75[.]188[.]83	@ip du C2 vidar de la campagne goldwing utilisé par cet implant dans ce rapport
Url	http://128.140.94.214	Url du C2 Vidar de la campagne lookat
IP	128[.]140[.]94[.]214	@ip du C2 Vidar de la campagne lookat
Url	http://5.75.142.250	Url du C2 Vidar de la campagne myfbook
IP	5[.]75[.]142[.]250	@ip du C2 Vidar de la campagne myfbook
