

Unpacking “Cruciferra”: An Analysis of a Sophisticated Crypter Service

 proofpoint.com/us/blog/threat-insight/unpacking-cruciferra-analysis-sophisticated-crypter-service

July 20, 2026

Unpacking “Cruciferra”: An Analysis of a Sophisticated Crypter Service



Chris Wakelin, Georgi Mladenov, Kyle Cucci, and the Proofpoint Threat Research Team

Key Findings

- Cruciferra is a sophisticated crypter service used by multiple unrelated cybercriminal threat clusters.
- It has been observed delivering a wide range of remote access trojans and infostealers.
- The malware employs extensive defense-evasion capabilities and over 90 variations of cryptographic functions to obfuscate its data and payloads.
- Proofpoint identified both production and apparent testing variants, indicating the service is under active development.

Overview

Proofpoint researchers are tracking Cruciferra, a crypter service that is used by multiple unrelated threat actors. Crypters are commonly used within the cybercriminal ecosystem to conceal malicious payloads, evade security controls, and improve malware delivery success rates. During our analysis, Proofpoint researchers identified both production and apparent testing samples, including variants containing debugging functionality and experimental features.

Cruciferra is written in [Mono](#) and features numerous techniques designed to evade detection, analysis, and incident response efforts. These include using indirect system calls, API and Import Address Table (IAT) unhooking, Bring-Your-Own-Vulnerable-Driver (BYOVD)-based EDR tampering, privilege escalation, persistence mechanisms, and a customized implementation of Process Ghosting used to execute payloads while minimizing forensic artifacts. The crypter also includes a notable emphasis on payload protection. Cruciferra supports a large collection of custom encryption routines, many of which appear to be dynamically assembled from components of established cryptographic algorithms. This approach creates significant variation between samples, complicating static analysis and signature-based defenses.

In this report, we'll look at Cruciferra's functionalities and observed real-world use. We also highlight the campaigns and malware families associated with the service, providing insight into a sophisticated and increasingly popular component of the modern malware-as-a-service ecosystem. Note that Cruciferra seems to be an “umbrella” name for a set of different crypters bundled together in the same service. In this report, we'll discuss the crypter we see the most often, which we simply call Cruciferra.

Cruciferra calls itself “the underground's most lethal crypter.” The malware is being advertised on Exploit[.]in, and likely on other forums as well. It was first made available for sale in the fall of 2025. Below is a screenshot from the forum exploit[.]in:

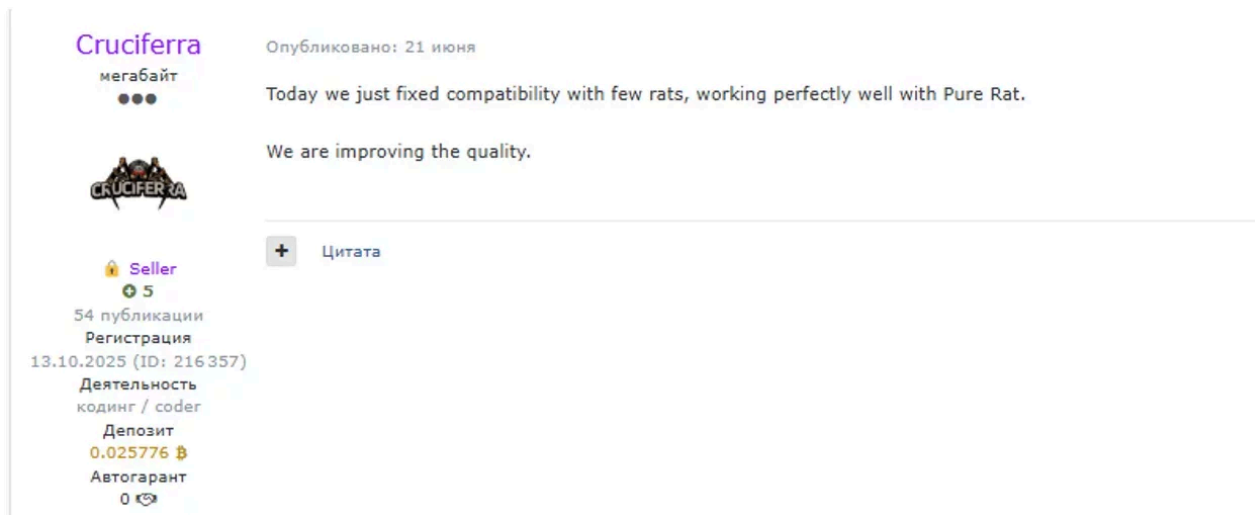


Figure 1: A public advertisement and notice of Cruciferra (from exploit[.in]).

The original posting of Cruciferra contained the following information (which has been truncated):

“Do you want to bypass Smartscreen, Chrome, Windows Defender and Kill AV Processes? SHARING ACCOUNT WITH FRIENDS ALLOWED

[...]

We allow free trials for members who have a reputation on the forum, i.e. long-time members. If you create an account today, we won't let you test it.

After payment, we do not offer any refunds; There is no chance of giving a refund. That's why we offer a two-day free trial for unique people.

If you want to try it for free before paying, we allow testing so you only pay if you want for the monthly package. Contact us.”

The malware author provides multiple tiers for sale; the more expensive the build is, the more features it has. Prices range from \$450 USD a month to \$2000 USD a month.

Campaign Details

Proofpoint has observed dozens of campaigns distributing commodity malware leveraging the Cruciferra crypter. In observed campaigns, malware is delivered via email, with Cruciferra used to obfuscate the ultimate payload. Cruciferra has the option of either dropping the payload to disk or downloading a payload from a staging server.

Researchers have observed Cruciferra delivered alongside multiple malware payloads such as zgRAT, AgentTesla, AsyncRAT, XLoader, XWorm, Phantom Stealer, Formbook, and Remcos.

Campaigns are conducted by multiple different threat actors. Targeting is opportunistic, and message volumes range from several hundred to thousands of messages per campaign. Although the targeting is opportunistic, some verticals were observed more frequently in campaigns, including financial services, healthcare, and government entities.

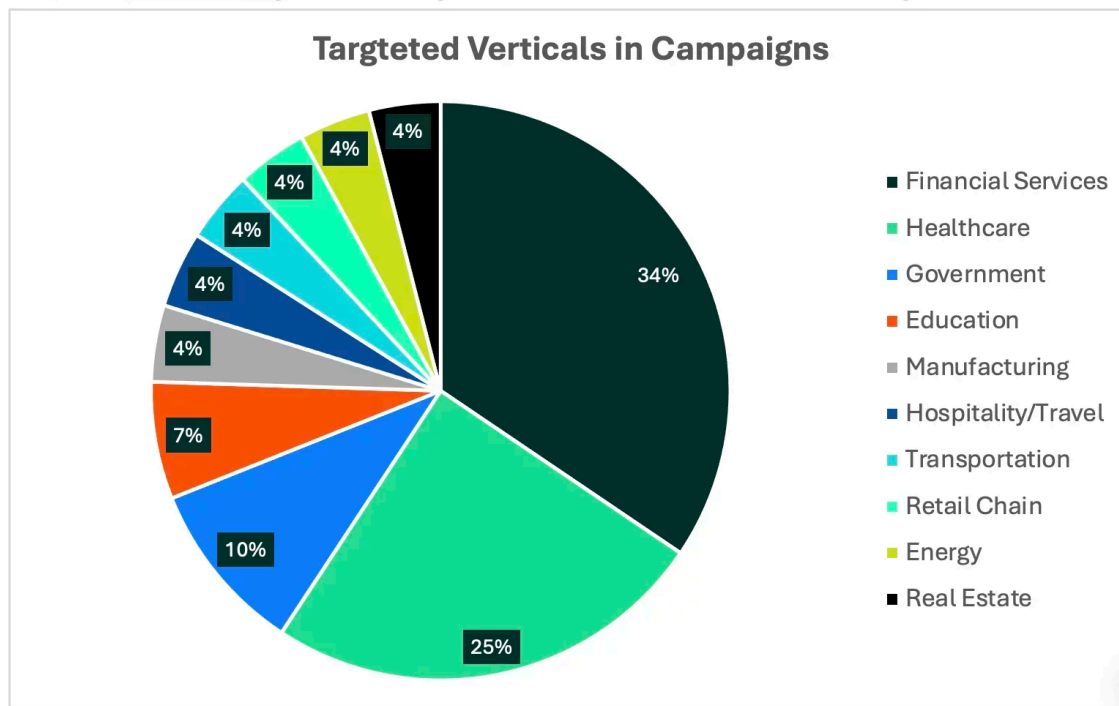


Figure 2: Observed verticals targeted in Cruciferra campaigns.

Example Campaigns: TA4922

Between late April and early June 2026, Proofpoint observed four campaigns attributed to Chinese-speaking cybercrime actor [TA4922](#) using Cruciferra to ultimately deliver AsyncRAT. Campaigns included up to 250 messages per campaign.

In each campaign, the actor leveraged tax-themed lures to drive victims to attacker-controlled landing pages hosting ZIP files containing an executable and DLL pair.

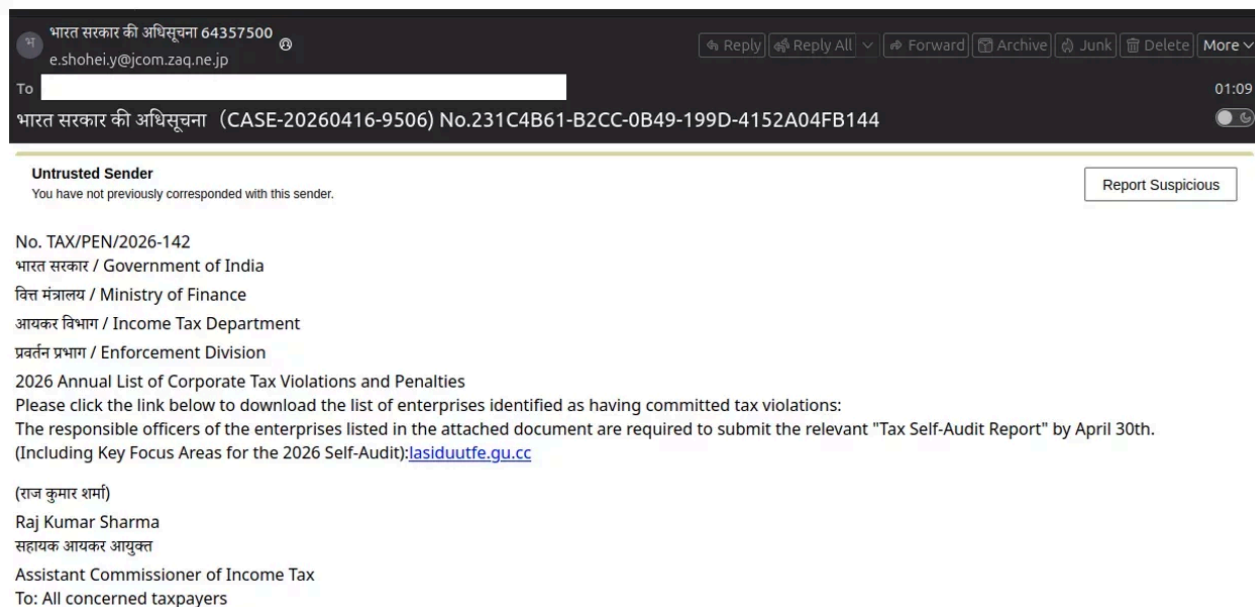


Figure 3: Tax-themed lure impersonating the Income Tax Department with an embedded URL.



Income Tax Department
Ministry of Finance, Government of India

No. TAX/
PEN/2026-142

भारत सरकार / Government of India
वित्त मंत्रालय / Ministry of Finance
आयकर विभाग / Income Tax Department
प्रवर्तन प्रभाग / Enforcement Division

Aayakar Bhawan,
New Delhi - 110001

OFFICE MEMORANDUM

Sub: कर अनुपालन की कमी और दंड सूचना / Tax Compliance Deficiency and Penalty Notice

- आपकी कंपनी की कर निरीक्षण के दौरान आयकर अधिनियम, 1961 की धारा 271(1)(c) के तहत कर संबंधी अनियमितताएं पाई गई हैं। The tax inspection of your company has revealed irregularities under Section 271(1)(c) of the Income Tax Act, 1961.
- इस सूचना की प्राप्ति के 72 घंटे (3 दिन) के भीतर निम्नलिखित दस्तावेज़ आयकर विभाग को प्रस्तुत करना अनिवार्य है। It is mandatory to submit the following documents to the Income Tax Department within 72 hours (3 days) of receipt of this notice.

 **आवश्यक दस्तावेज़ सूची / Required Documents List**
नीचे दिए गए लिंक से दस्तावेज़ डाउनलोड करें / Download documents from the link below

 **दस्तावेज़ डाउनलोड करें / Download Documents**

- निर्धारित समयवधि में दस्तावेज़ प्रस्तुत न करने पर आयकर अधिनियम की धारा 276C के तहत कानूनी कार्रवाई की जाएगी। Legal action will be taken under Section 276C of the Income Tax Act if documents are not submitted within the stipulated time.

(राज कुमार शर्मा)
Raj Kumar Sharma
अध्यक्ष, प्रवर्तन प्रभाग

Figure 4: Fake Income Tax Department portal used to host the ZIP file which initiates the Cruciferra infection chain.

The email and PDF lures followed consistent social engineering patterns while impersonating government tax authorities through “Income Tax Department” or “Government of India” notifications. The landing pages, which have been observed in numerous prior TA4922 campaigns, were designed to closely mimic legitimate government tax portals and prompt the recipient to download required documents which reinforce legitimacy and urgency. The URLs leading to these landing pages were delivered either directly within the email body or via PDF attachments containing embedded links.

Example Campaign: XWorm

[Tax](#) and government-related themes are frequent favorites of cybercriminals, and the U.S. Social Security Administration (SSA) is often abused in malware campaigns, including from actors using Cruciferra. For example, in May, Proofpoint researchers observed emails impersonating the SSA regarding tax documents. (Curiously, the emails referred to items that needed to be completed by January 2026; it's possible the actor repurposed an old lure, or mistakenly included the wrong date.)

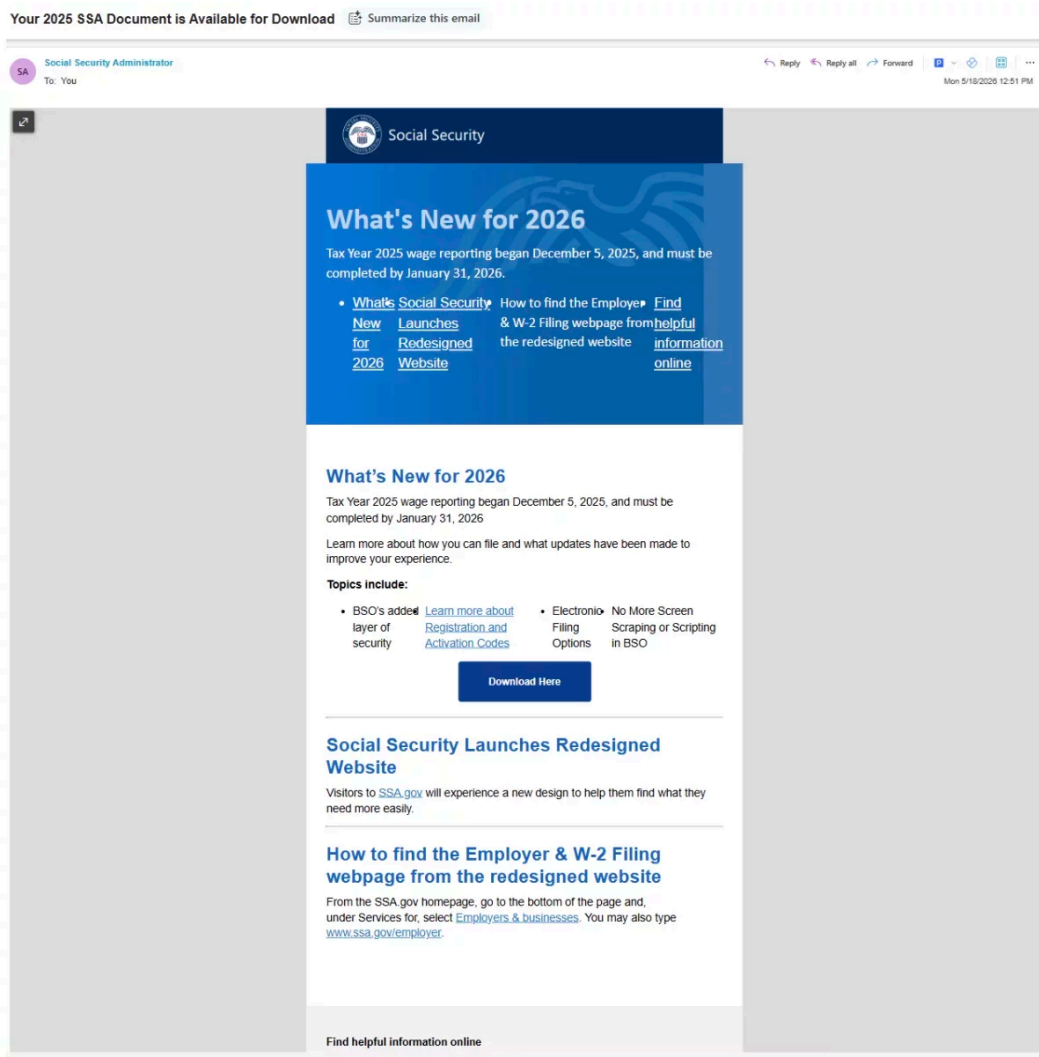


Figure 5: Fraudulent SSA emails.

These messages contained URLs leading to the download of a VHD file which, if clicked, ran an executable which ran Cruciferra. This malware then led to XWorm and AdaptixC2. Researchers at Deception.Pro [also published details](#) on this campaign.

Example Campaign: Guest Complaint

In a Cruciferra campaign observed at the end of June, threat actors leveraged email themes related to bed bugs and guest complaints to target organizations in the hospitality and travel industries.

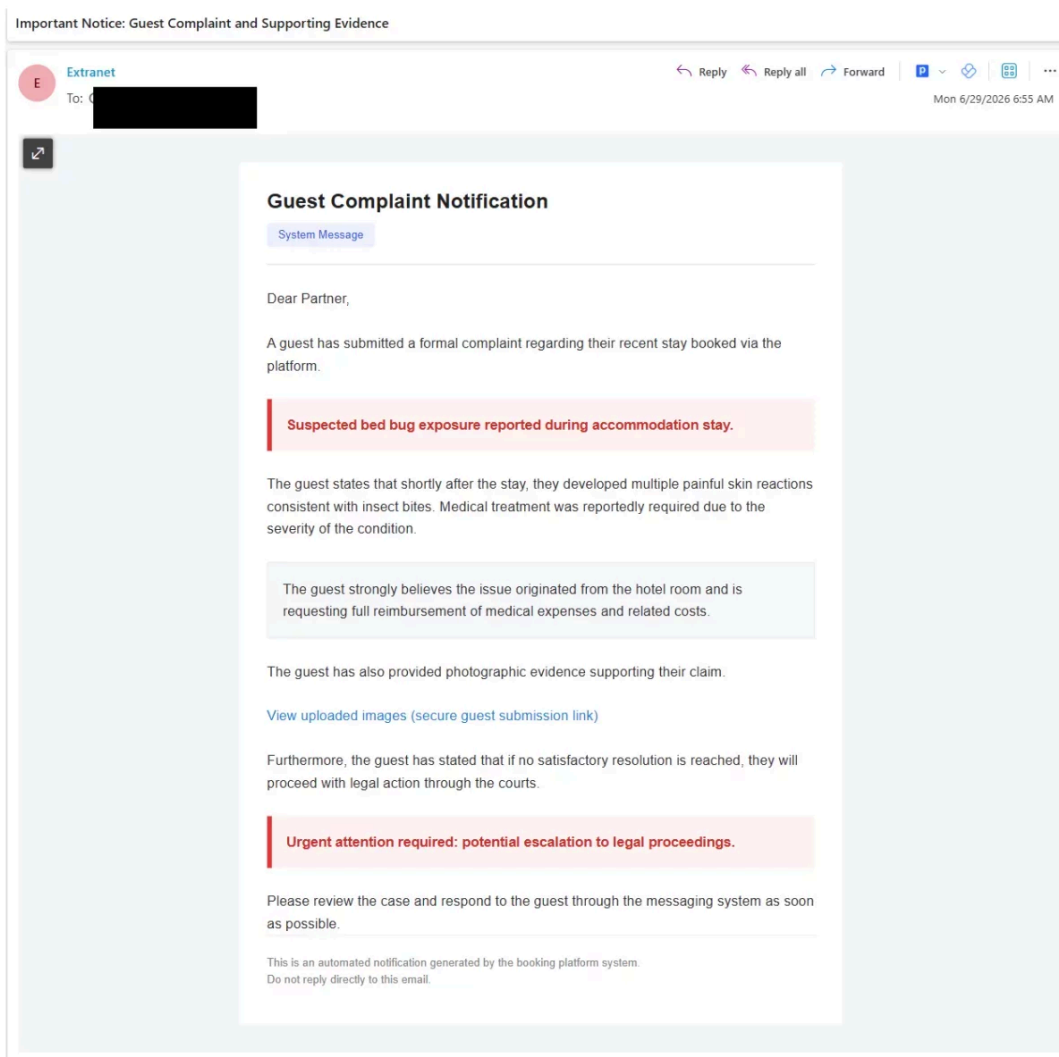


Figure 6: Guest complaint lure.

These messages contained URLs masquerading as links to evidence provided by a guest, but led to the download of a zipped LNK file that launched a PowerShell command, which then executed a PowerShell script. This script first fingerprinted the user's system and reported the collected information to an actor-controlled server. It then downloaded a ZIP archive which ultimately led to the installation of Cruciferra. Cruciferra was observed loading zgRAT.

Cruciferra Analysis

Based on our observations, Cruciferra is always executed via DLL side-loading. The infection chain involves a ZIP or similar archive file that contains an executable and a DLL. When the target runs the executable file, the DLL (which contains Cruciferra's code) is side-loaded, and the executable invokes the main malicious function code inside the DLL. Cruciferra's primary purpose is to ensure the target system isn't a sandbox or malware analyst's virtual machine before dropping and executing the payload.

Evasion and Anti-Analysis Techniques

Before the payload is deployed on the victim system, Cruciferra uses several techniques to detect and evade endpoint defenses, sandboxes, and malware analysts.

Decoy Exported Functions

Cruciferra DLLs contain many fake exported functions that point to junk code or do nothing. In some cases, the Cruciferra DLL can contain hundreds or even thousands of exports, most of them pointing to junk code. One or a select few functions contain a call instruction that jumps to the "real" malicious code. These decoy exported functions are an anti-sandbox and anti-analysis measure that makes it more difficult to reach the actual malicious code.

Name	Address	Ordinal
TlsCallback_0	000000001801A0CC8	
DllEntryPoint	000000001801A0C88	[main entry]
_01yTEQBjJ0A4yCXewWb2z	0000000018005EB50	1
_02WGQVDMtYxp2jIf8GHITwqlYHW	0000000018005D7F0	2
_03xBDz1J9kaOGFxdJR8YnmNkdAZJ0t	0000000018004EE90	3
_048o2f8vYr4WA8ObBbrhJyP	00000000180051C30	4
_07t5T62Myb0kt6Mpphq3DhKa3zM	0000000018005BDB0	5
_093XBjL89NbdbgBrFeTN7R8cVJxd	00000000180053AF0	6
_09ixI8NQ	00000000180050990	7
_09vgd7bIug6YykDnEiKs26Tn0WGu	0000000018005C1B0	8
_0AVf2Wppf25jhpW3znRiDy	0000000018004B170	9
_0DdaeH5	00000000180052A30	10
_0Ft6J2TE9ggv	0000000018005C2F0	11
_0HaA2iY	000000001800565B0	12
_0JBAh8bFNEdNLgh6amn4710	00000000180055A70	13
_0JniAK9	000000001800575F0	14
_0JzMU6rk1mA2	000000001800583D0	15
_0NQBeg5kmdrzJktHu2vSi	0000000018005C350	16
_0OjO6efdfFEAwmlow1yp	0000000018005C330	17
_0PcqecpVh	00000000180048CD0	18
_0PymgONzTpyuWvDWctdESw5Ibp	000000001800567B0	19
0Qq53vizncc2L1AhrS74n3Xqpur	000000001800524D0	20

Figure 7: IDA Pro snippet showing a typical list of exported functions in a Cruciferra sample.

Console Window Hiding

Cruciferra uses a unique technique to prevent console windows from spawning from its processes (which could alert the user to an infection). The malware hides console windows by spawning a background thread that loops 100 times (while sleeping 50ms between iterations), and walking up the process tree from the current process. At each level, it calls EnumWindows to find all windows belonging to that process ID, checks if the window class is "ConsoleWindowClass", and if so, hides it using both ShowWindow(SW_HIDE) and SetWindowPos(SWP_HIDEWINDOW). The persistent retry loop ensures any console windows that appear after startup (e.g., from cmd [.]exe or conhost [.]exe parents) are caught and hidden quickly.

Function Unhooking

Cruciferra attempts to unhook several functions. A hook is a small piece of code inserted into Windows API functions that allows EDR, antivirus (AV), and sandbox solutions to monitor function calls and analyse a program's behaviour. By removing or modifying these hooks, Cruciferra uses an evasion technique known as unhooking to reduce visibility into its activities and hinder detection by endpoint defences and analysis tools:

```

v14 = String::decrypt(&stru_180306AA0); // "CoreInit: Starting UnhookDll..."
debug_output(v14);
v15 = String::decrypt(&stru_180307C48); // "ntdll.dll"
UnhookDll(v15);
v16 = String::decrypt(&stru_180308648); // "kernel32.dll"
UnhookDll(v16);
v17 = String::decrypt(&stru_180309048); // "advapi32.dll"
UnhookDll(v17);
v18 = String::decrypt(&stru_180308118); // "ole32.dll"
UnhookDll(v18);
v19 = String::decrypt(&stru_180307AE8); // "shell32.dll"
UnhookDll(v19);
v20 = String::decrypt(&stru_180309200); // "combase.dll"
UnhookDll(v20);
v21 = String::decrypt(&stru_1803080E0); // "oleaut32.dll"
UnhookDll(v21);
v22 = String::decrypt(&stru_180306D88); // "CoreInit: UnhookDll loop complete."
debug_output(v22);

```

Figure 8: Cruciferra unhooking several DLLs to inhibit monitoring.

```

v4 = String::decrypt(&stru_1802FD968); // "\KnownDlls\"
v5 = String::concat(v4, a1); // e.g. "\KnownDlls\ntdll.dll"
...
NtOpenSection(&v35, 4, v36);
...
NtMapViewOfSection(v35, -1, &v34, 0, 0, 0, &v33, 2, 0, 2); // Map the clean copy of the original DLL on disk into memory
...
if (*v15 == '.' && v15[1] == 't' && v15[2] == 'e' && v15[3] == 'x' && v15[4] == 't') // Locate the .text section (executable code) in original DLL and parse
    break;
...
for (i = 0; i < v17; ++i) // Overwrite hooked code with clean code
    data_dst[i] = data_src[i];
...

```

Figure 9: Cruciferra unhooking code (simplified)

BYOVD-based Evasion

Cruciferra attempts to disable endpoint defenses like EDR by dropping a vulnerable “helper” driver as a BYOVD (Bring-Your-Own-Vulnerable-Driver) technique. This is a well-documented technique, and involves abusing a vulnerable kernel driver to issue low-level commands (or, I/O Controls - IOCTLs) to the operating system to terminate target processes to prevent detection.

In this case, Cruciferra often abuses the legitimate (but vulnerable) GoFlyDrv.sys driver and attempts to terminate various EDR processes by looping through the running process list, identifying suspect process names, and invoking DeviceIOControl to send process termination codes to the suspected processes.

Alternatively, in place of the GoFlyDrv.sys driver, the following helper drivers may be dropped and used as part of the BYOVD technique. There are likely others as well.

Driver Name	Driver Hash
Core64.sys	17aae57cf6255c7eb169bf62ea67376d9708976eb7831f8cdd0ea38bdcb37dc4
GoFlyDrv.sys	2fd added 13a0c548bb68c9d5aa8599a9265d4659da3e237fe7a42ac6ac06b9a06a
HwOs2Ec.sys	c4e93449453cf67c5d5605bb8f425207a738a242fdb432d720acc32faa74926c
LnVMSRIO.sys	c5b1e9aafc8f2b4ab05effc00fd43f3114b9ef1d592a086c952793ac4e299809
MemoryInformer.sys	7887e919555fb5948c217556ba149392a72982b1bc427d3db779db9dcbf09ee8

NTIOLib_X64.sys	09bedbf7a41e0f8dabe4f41d331db58373ce15b2e9204540873a1884f38bdde1
ProcessMonitorDriver.sys	5b4f59236a9b950bcd5191b35d19125f60cfb9e1a1e1aa2e4f914b6745dde9df
selfprot.sys	c46e907886e2158cbc453e767183aecf07887b5ac8848f19684451883d69f5f0

Indirect Syscalls

Cruciferra uses indirect syscalls to evade endpoint defenses, sandboxes, and analysis tooling. To do so, the malware reads a clean copy of ntdll.dll on disk and stores all stub pointers in a global structure for later usage. This allows the rest of the malware to call these APIs while bypassing EDR and sandbox inline hooks on ntdll.dll. The selected syscalls are:

- NtProtectVirtualMemory
- NtCreateSection
- NtMapViewOfSection
- NtCreateTransaction
- NtRollbackTransaction
- NtOpenKey
- NtSetValueKey
- NtClose
- NtSetInformationFile

IAT Unhooking

Cruciferra repairs the IAT (Import Address Table) to remove any IAT hooks that endpoint defenses, a sandbox, or analysis tools may have implemented. IAT hooking is an older technique and not often used anymore, so it's notable that Cruciferra implements this unhooking technique. Below, you can see a snippet of the IAT repair/unhooking code:

```

if ( ProcAddress_0 )
{
    if ( *v14 != ProcAddress_0 )                // Only patch if it differs (hooked)
    {
        v27[0] = v14;
        *(_QWORD *)&v26 + 1) = 8LL;
        NtProtectVirtualMemory = objSYSAPI.NtProtectVirtualMemory;
        if ( NtProtectVirtualMemory )
        {
            p_enter(&v24);
            v20 = NtProtectVirtualMemory(
                (HANDLE)0xFFFFFFFFFFFFFFFFLL,    // current process
                v27,                             // &address
                (char *)&v26 + 8,               // &size (8 bytes)
                4u,                             // PAGE_READWRITE
                (PULONG)&v26);                  // old protect
            p_exit(&v24);
            if ( v20 >= 0 )
            {
                *v14 = ProcAddress_0;           // OVERWRITE the IAT entry
                // Restore original page protection
                v27[0] = v14;
                *(_QWORD *)&v26 + 1) = 8LL;
                NtProtectVirtualMemory = objSYSAPI.NtProtectVirtualMemory;
                v23 = v26;                       // old protect value
                p_enter(&v24);
                NtProtectVirtualMemory((HANDLE)0xFFFFFFFFFFFFFFFFLL, v27, (char *)&v26 + 8, v23, (PULONG)&v25);
            }
        }
    }
}

```

Figure 10: Snippet of IAT unhooking code in Cruciferra.

Disabling User Notifications

To maintain stealth on the infected endpoint, Cruciferra modifies a few registry keys to disable user notifications. These are as follows:

Registry Key	Registry Value	Description
Software\Microsoft\Windows\CurrentVersion\PushNotifications	ToastEnabled	Disabling "Toast" notifications prevents Windows notification pop-ups from appearing. May suppress notifications from Windows Defender, SmartScreen, etc.
Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced	Balloon	Suppresses classic notification balloons such as from Security Center.
Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced	ShowInfoTip	Disables infotips (when hovering over files/folders).

Privilege Elevation and Persistence

Cruciferra checks if it is running with Administrator privileges, and if not, attempts to elevate its privileges by bypassing UAC using the COM Elevation Moniker. We won't cover this technique in this blog post, but you can read more about this here: <https://learn.microsoft.com/en-us/windows/win32/com/the-com-elevation-moniker>.

Additionally, Cruciferra establishes persistence by writing to the registry *Software\Microsoft\Windows\CurrentVersion\Run* key with a default value of "putty". This ensures Cruciferra runs after system reboot.

Decrypting the Payload

Payloads are stored in Cruciferra's ".reloc" section in the following format:

```
struct PAYLOAD_STORAGE {
    char    egg[]; // varies "_qDevon", "MuchInfo", etc.
    DWORD   size;
    byte    data[size]; // encoded
}
```

Figure 11: Payload storage format in the binary.

Payloads are encoded in the binary using Base16 encoding with a custom character set - "PQRSTUVWXYZ[]^_":

```
def decode(data):
    ndata = len(data)
    result = bytearray(ndata >> 1)
    for i in range(ndata >> 1):
        hi = (data[2 * i] - 0x50) & 0xF
        lo = (data[2 * i + 1] - 0x50) & 0xF
        result[i] = (hi << 4) | lo
    return result
```

Figure 12: Part of the Base16 decoding routine in Cruciferra.

After the decoding stage, the payloads, along with the helper drivers (the BYOVD driver(s) mentioned previously), are stored in a simple file structure:

```
struct FILESYSTEM {
    DWORD          number_of_files;
    FILE_ENTRY     fileset[number_of_files];
};

struct FILE_ENTRY {
    byte           filename_size;
    char           filename[filename_size];
    DWORD          data_size;
    byte           data[data_size];
}
```

Figure 13: Payload file structure.

The filenames are used only internally, and they vary from a set of random words stitched together to human-unreadable characters (possibly UTF16 encoded).

The majority of the Cruciferra samples Proofpoint threat researchers have analysed store their payloads in the manner just outlined, but we have seen variants of Cruciferra loading their payloads from a separate file (which is dropped to disk alongside the original delivery package, such as inside the original ZIP archive). We've also seen Cruciferra samples that download their payloads from a staging server. These variants seem to be less common, however.

Bring Your Own Crypto Algorithms

One of the most interesting features of Cruciferra is the number of unique encryption algorithms used to encrypt and decrypt the stored payloads. Cruciferra can use one of a over 90 different encryption algorithms. Notably, these algorithms are almost never "complete" algorithms. Each is modified or pieced together from multiple other algorithms. As an abstract example, Cruciferra may use the KSA (Key Scheduling Algorithm) from "Algorithm A" and then borrow the block cipher code from "Algorithm B", forming a new custom algorithm. The algorithm used to encrypt payloads and strings in each set of samples is different, and there is such a large variance of these algorithms, which means it is probably randomly generated (polymorphically) from elements of well-known hashing, PRNG, and cipher algorithms.

Here are some examples of the algorithms from which Cruciferra derives components:

- Keccak (<https://en.wikipedia.org/wiki/SHA-3>)
- Cyclic (Circular) Convolution (https://en.wikipedia.org/wiki/Circular_convolution)
- Generalized Feistel and Square-OR Feistel (https://en.wikipedia.org/wiki/Feistel_cipher)
- SPECK-128/256 CTR (<https://github.com/Naruto/simon-speck-c>)
- Multiply-accumulate PRNG (https://en.wikipedia.org/wiki/Multiply-with-carry_pseudorandom_number_generator)
- Modified Threefish-256 CTR (<https://en.wikipedia.org/wiki/Threefish>)
- Squares+Xorshift64 PRNG (<https://en.wikipedia.org/wiki/Xorshift> and https://en.wikipedia.org/wiki/Middle-square_method)
- Various ARX (Addition–Rotation–XOR)-based algorithms (2-layer ARX, Serial cascade ARX, Sponge ARX, Multiple-mod ARX, etc.) (https://en.wikipedia.org/wiki/Rotational_cryptanalysis)
- DES-CBC-PKCS7 (one of the few fully standard algorithms Cruciferra uses).

Loading the Payload (Tweaked Process Ghosting)

Finally, to load its payload into memory, Cruciferra employs a variant of Process Ghosting. Process Ghosting is when malware creates a temporary file, marks it for pending deletion via `NtSetInformationFile`, writes the malicious payload into it, then creates an image section (`NtCreateSection` with `SEC_IMAGE`) from that file. Once the file handle is closed, the operating system deletes the file from disk while the section persists in memory. A legitimate process is then created in a suspended state, the ghost section is mapped into it via `NtMapViewOfSection`, the thread context is redirected to the payload's entry point, and the thread is resumed. The result is a running process backed by a PE image that never existed on disk in a scannable state.

However, Cruciferra adds an additional layer to the Process Ghosting technique by adding two anti-EDR techniques. First, Cruciferra patches `ZwQueryVirtualMemory` hooks so when endpoint defenses such as EDR queries memory via `ZwQueryVirtualMemory` on the ghosted memory region inside that process, the hook intercepts the call and returns a sanitized result. This hides the fact that the backing file is deleted or anomalous.

Second, Cruciferra attempts to neuter the `NtManageHotPatch` function, which can be used by the operating system to apply live patches to running processes, but can also be used to validate the integrity of loaded image sections against their on-disk backing files. This will cause `NtManageHotPatch` to either return immediately or return an error, preventing the kernel's hot-patching function from inspecting or validating the ghosted image.

```
// Resolves the target function in the remote process's ntdll.dll
v7 = sub_18001A280(v4, v6); // Find export address
v6 = String::decrypt(&stru_180301610); // "NtManageHotPatch" (encrypted)
...

// Reads the first 32 bytes of the function from the remote process:
if ( !sub_180019D60(a1, v7, (__int64)v8, 0x20uLL, (SIZE_T *)&v33 + 1) || DWORD2(v33) != 32 )
...
// Locates the syscall instruction, prepares a 6-byte patch payload, and overwrites the function's
// syscall in the remote process:
v5 = Buffer::new(&qword_180263970, 6uLL);
*(_DWORD *)v5->data = patch; // Patch bytes
*((WORD *)&v5->data[4] = *((WORD *)&patch + 2);
```

Figure 14: Snippet of Cruciferra's `NtManageHotPatch` hooking code.

We have observed the following payloads being dropped by Cruciferra:

- Agent Tesla
- AsyncRAT / DCRAT
- DarkCloud Stealer
- Formbook & XLoader
- PhantomStealer
- Remcos
- Snake Keylogger
- ValleyRAT (also known as Winos4.0)
- XWorm
- zgRAT

Tracking Cruciferra

Finally, we noticed an interesting trend in many Cruciferra-packed samples. The File Version Information section of the PE files contains randomly generated Copyright, Product, Description, and other fields that follow a standard format. The Copyright metadata, for example, may be a date followed by two to four random words, such as "2026 Colpoplastic Semipractical Group". The Product and Description fields are similar, containing a string of random words. Here are some examples from samples in VirusTotal:

File Version Information	
Copyright	© 2026 Colpoplastic Semipractical Group
Product	Serriedly Coom
Description	Tanny barycentric vanillery retroflexion browner pliant celibatory unbeliefs subfastigate.
Original Name	SplinedStulty.exe
Internal Name	Isonuclear Cive
File Version	7.4.470.84

Figure 15: Cruceferra PE metadata example 1.

File Version Information	
Copyright	© 2026 Doncella Inc.
Product	Newspaperwoman Blender
Description	Amity tachymetry moats smeddums facinorous tempers.
Original Name	VaingloryMandament.exe
Internal Name	Enacture Microcyte
File Version	6.90.856.39

Figure 16: Cruceferra PE metadata example 2.

File Version Information	
Copyright	© 2025 Bespete Ltd.
Product	Lewing Theogonies
Description	Shipmates insurrectionism formes paternalness spongelet unassimilated rappite fatil.
Original Name	PapyrineCyrtomium.exe
Internal Name	Signatural Unescheatable
File Version	1.26.106.18

Figure 17: Cruceferra PE metadata example 3.

Using Yara signatures, metadata formatting, and other indicators, we can track uploads of Cruciferra in VirusTotal. If the compile and debug timestamps are correct (we have no reason to believe they are falsified) we can see new builds of the crypter uploaded very regularly, showing ongoing crypter service operations. The following metadata from VirusTotal shows that on 9 July 2026 a new sample was packed with Cruciferra every few minutes.

Thu	Jul	9	02:53:01	2026
Thu	Jul	9	02:54:59	2026
Thu	Jul	9	03:11:36	2026
Thu	Jul	9	03:13:35	2026
Thu	Jul	9	03:16:14	2026
Thu	Jul	9	03:17:57	2026
Thu	Jul	9	03:19:33	2026
Thu	Jul	9	03:20:12	2026
Thu	Jul	9	03:21:34	2026
Thu	Jul	9	03:23:12	2026
Thu	Jul	9	03:26:48	2026
Thu	Jul	9	03:29:19	2026
Thu	Jul	9	03:32:33	2026
Thu	Jul	9	03:33:22	2026
Thu	Jul	9	03:34:10	2026

Figure 18: Debug timestamps metadata of Cruciferra-packed samples in VirusTotal.

Conclusion

While crypters have long been used to evade detection and increase malware delivery and execution success rates, Cruciferra distinguishes itself through its extensive and unique defense-evasion capabilities, modular design, and highly customized and varied approach to payload protection.

During our investigations, we observed Cruciferra delivering numerous malware families, including various remote access trojans and infostealers, highlighting its role as an enabling technology within the cybercrime ecosystem. Proofpoint will continue to monitor the development and adoption of Cruciferra and provide updates as new capabilities and campaigns are identified.

Example IOCs

Below are examples of Cruciferra and related indicators that were part of our investigation:

IOC	Description	First Seen
hxxp://hsahyteiows[.]gu[.]cc	TA4922 Cruciferra / AsyncRAT Payload URL	28 April 2026
hxxp://yicoweytcbtw[.]gu[.]cc	TA4922 Cruciferra / AsyncRAT Payload URL	28 April 2026
hxxp://nciyeyrawoe[.]gu[.]cc	TA4922 Cruciferra / AsyncRAT Payload URL	28 April 2026
hxxp://lasiduutf[.]gu[.]cc	TA4922 Cruciferra / AsyncRAT Payload URL	28 April 2026
3c181f642e24c28602a87be7f195e2f3d1ffa30b37e20f5121d99f88b22ab80e	TA4922 Cruciferra / AsyncRAT SHA256 Tax-Number52563.zip	28 April 2026
hxxp://xkcifgieusr[.]gu[.]cc	TA4922 Cruciferra / AsyncRAT Payload URL	4 May 2026
hxxp://viuyeyrwqs[.]gu[.]cc	TA4922 Cruciferra / AsyncRAT Payload URL	4 May 2026
hxxp://pmcjsuyraw[.]gu[.]cc	TA4922 Cruciferra / AsyncRAT Payload URL	4 May 2026

hxxp://laiwutrencr[.]gu[.]cc	TA4922 Cruciferra / AsyncRAT Payload URL	4 May 2026
hxxp://maisytawe[.]gu[.]cc	TA4922 Cruciferra / AsyncRAT Payload URL	4 May 2026
hxxp://kawosyetw[.]gu[.]cc	TA4922 Cruciferra / AsyncRAT Payload URL	4 May 2026
hxxp://nviuawusye[.]gu[.]cc	TA4922 Cruciferra / AsyncRAT Payload URL	4 May 2026
hxxp://faeytrdeaw[.]gu[.]cc	TA4922 Cruciferra / AsyncRAT Payload URL	4 May 2026
hxxp://figyuyrqrw[.]gu[.]cc	TA4922 Cruciferra / AsyncRAT Payload URL	4 May 2026
hxxp://hfyuayustrv[.]gu[.]cc	TA4922 Cruciferra / AsyncRAT Payload URL	4 May 2026
hxxp://jsiruytrawey[.]gu[.]cc	TA4922 Cruciferra / AsyncRAT Payload URL	4 May 2026
hxxp://kawuuterta[.]gu[.]cc	TA4922 Cruciferra / AsyncRAT Payload URL	4 May 2026
hxxp://nvsieyrrawe[.]gu[.]cc	TA4922 Cruciferra / AsyncRAT Payload URL	4 May 2026
66dbe675480dc229e5b3ab8ad74207f73486e64e57805074f784bb2e01bcb865	TA4922 Cruciferra / AsyncRAT SHA256 Tax-Number809863.zip	4 May 2026
hxxp://fuaytrwese[.]love	TA4922 Cruciferra / AsyncRAT Payload URL	28 May 2026

hxxp://qeusasytua[.]love	TA4922 Cruciferra / AsyncRAT Payload URL	28 May 2026
hxxp://svuatwea[.]love	TA4922 Cruciferra / AsyncRAT Payload URL	28 May 2026
hxxp://vusuydryt[.]love	TA4922 Cruciferra / AsyncRAT Payload URL	28 May 2026
hxxp://xnbscuya[.]love	TA4922 Cruciferra / AsyncRAT Payload URL	28 May 2026
hxxp://ncduuyese[.]live	TA4922 Cruciferra / AsyncRAT Payload URL	28 May 2026
hxxps://oakwusya[.]love	TA4922 Cruciferra / AsyncRAT Payload URL	28 May 2026
hxxp://syfiaydytea[.]live	TA4922 Cruciferra / AsyncRAT Payload URL	28 May 2026
a6fb779be35592fb0ff624a8f8e12ab3cafe7bcfc312cd98263814db7fb01e02	TA4922 Cruciferra / AsyncRAT SHA256 Tax-Number119863.zip	28 May 2026
59ad96dd3b4d5f10a5c53bbd465446e52dc7701a4ac633632f762bf1336d3347	TA4922 Cruciferra / AsyncRAT SHA256 Tax-Number101863.zip	28 May 2026
hxxp://jaiydteds[.]love	TA4922 Cruciferra / AsyncRAT Payload URL	1 June 2026
hxxp://mksfuuerwo[.]live	TA4922 Cruciferra / AsyncRAT Payload URL	1 June 2026
hxxp://fiusyevr[.]live	TA4922 Cruciferra / AsyncRAT Payload URL	1 June 2026

hxxp://lisiutegrm[.]live	TA4922 Cruciferra / AsyncRAT Payload URL	1 June 2026
hxxp://paiwudyeyea[.]love	TA4922 Cruciferra / AsyncRAT Payload URL	1 June 2026
hxxp://xuastyrdqk[.]love	TA4922 Cruciferra / AsyncRAT Payload URL	1 June 2026
hxxps://fvxcuvuyte[.]live	TA4922 Cruciferra / AsyncRAT Payload URL	1 June 2026
hxxps://kdsuyrse[.]live	TA4922 Cruciferra / AsyncRAT Payload URL	1 June 2026
hxxps://hsauyeet[.]live	TA4922 Cruciferra / AsyncRAT Payload URL	1 June 2026
6dbd6f9f2fa636c16ac4fa81418b68a604424861b9650dd9c4f2b0ba6f67d6ac	TA4922 Cruciferra / AsyncRAT SHA256 Tax-Number33863.zip	1 June 2026
hxxps://almacensantangel[.]com/wp-includes/assets/YourSSA_Documents_0000000676152_05_187_2026_Document_0000000676152[.]rar	Cruciferra / XWorm Payload URL	18 May 2026
gatuso[.]duckdns[.]org	XWorm C2	18 May 2026
3f31aee0948d16f8d64bf6bec69a4331099993e502b11bfc56b2c0112024489d	Cruciferra / zgRAT Payload SHA256 photo295825092412.zip	29 June 2026
hxxps://digital-magicians[.]com/photo295825092412[.]zip?_r=ea623202	Cruciferra / zgRAT Payload URL	29 June 2026
0zbqnac1t4dv2t2wuodv1m[.]com	zgRAT C2	29 June 2026

[Previous Blog Post](#)

Subscribe to the Proofpoint Blog

Business Email:

© 2026. All rights reserved.