Latrodectus Affiliate Resumes Operations Using Brute Ratel C4 Post Operation Endgame

By Binary Analysis

Published: 2024-06-24 · Archived: 2026-04-06 00:54:26 UTC

Executive Summary

RevEng.AI observed a *Latrodectus* sample (a.k.a. *Unidentified 111*, *Lotus*, *BLACKWIDOW*, *IceNova*) delivery chain using a malicious JavaScript (JS) stager uploaded to a third-party public malware scanning service on 23 June 2024. Since early January until the present, RevEng.AI has observed versions 1.1 to 1.3 in operational use by the adversary (although earlier versions do exist, as documented by industry reporting [1]).

Latrodectus is a loader typically delivered by phishing emails containing a shortened link to a likely compromised captcha-gated WordPress website, masquerading as legitimate services such as Cloudflare. The use of a captcha-gated delivery website is likely to circumvent automated payload retrieval. The user is prompted to download and execute an MSI (Microsoft Windows Installer), and in some cases, the delivery chain also contains a PDF to socially engineer the user. The malware borrows code heavily from the open-source BlackLotus malware [2], which contains a ready-to-use bot and command-and-control (C2) component.

Latrodectus was one of the many malware families targeted by Operation Endgame, an international law enforcement effort against malware loader infrastructure [5]. Since the operation, *Latrodectus* has quickly rebuilt its infrastructure and returned to its standard mode of operation.

This blog post covers the most recent distribution chain, noteably the use of Brute Ratel, and the anti-analysis methods employed by the malware.

Latrodectus Delivery Chain

1. JavaScript - Download & Execute MSI

The JavaScript-based stager (SHA-256: `3ac8decd825d1ed7a86ed86d7789b44c0f0c467d4f482ab0863df1b7b1e3e8cc`), named `Form_Ver-11-58-52.js`, for this distribution campaign is configured with the `msiPath` of `http[:]//85.208.108[.]63/BST.msi`. The JS file is consistent with previous campaigns used to download and execute *Latrodectus* MSIs. The `Installer.InstallProduct` API is used [3], which natively handles the download and execution of local and remote MSIs (HTTP, UNC path, etc.).

2. MSI Package

The MSI creates a child process to decompresses the contents of a bundled CAB archive (SHA-256: `c89d15789fd9e0b23e62bbf038d2ddbcea5618573517f3382790b4b0434933df`) named `disk1.cab` to `%APPDATA%`.

The CAB file contains the malicious DLL file `aclui.dll` (SHA-256: `c5dc5fd676ab5b877bc86f88485c29d9f74933f8e98a33bddc29f0f3acc5a5b9`), which is a packed Brute Ratel C4 (BRC4) badger implant. The DLL export `edit` is then executed using `rundll32.exe` by the MSI installer, i.e: `rundll32.exe C:\Users\Ivan\AppData\aclui.dll,edit`.

Although Brute Ratel is a legitimate red-teaming tool, it has been previously used by threat actors [4].

3. BRC4 Badger

After BRC4 unpacks itself (SHA-256: `0d3fd08d237f2f22574edf6baf572fa3b15281170f4d14f98433ddeda9f1c5b2`) and self-injects, a long sleep is likely used to avoid immediate execution by automated sandbox solutions. The C2 configuration for the BRC4 badger can be observed below in Table 1.

Key	Value	Channel
C2	`barsen[.]monster:7444` `kurvabbr[.]pw:7444`	HTTPS

Table 1 - Badger Configuration

The configuration hosts, at the time of writing, resolve to `94.232.249[.]86`.

4. *Latrodectus 1.3* via BRC4

The BRC4 badger is configured to execute a *Latrodectus* payload upon a victim connecting. The *Latrodectus* core bot component (`dbd85d5dd501bb7fad3990f0801d32da438a5bc60bd7cf6999d5bc535291146c`) is injected into the common target `explorer.exe`. The full configuration of this *Latrodectus* sample is avaliable in the *Latrodectus Configuration* section.

The stealer module is also downloaded and injected into the current process, `explorer.exe`, (SHA-256: `44ccc3fbd3e15e8bdb063616d9baa37b1f9ab9121759fd467c943b7611860f72`), targeting Microsoft Edge, Microsoft Internet Explorer, Microsoft Outlook, Firefox, Google Chrome, 360 Browser, Yandex Browser, and more.

RevEng.AI AI binary code similarity engine was able to quickly recover SQLite debug symbols within the stealer module and aid in the reverse-engineering process:

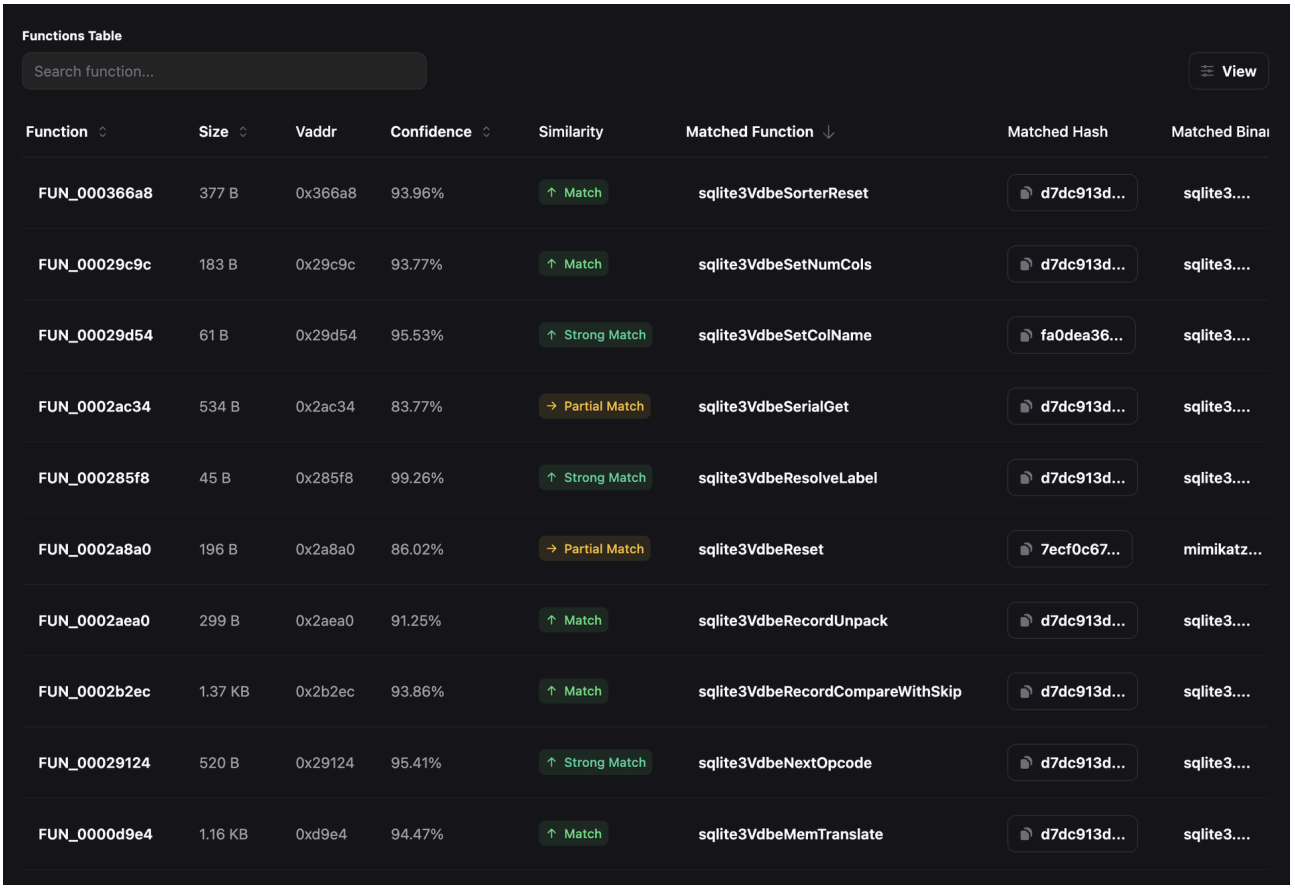

Figure 1 - SQLite Symbol Recovery

Anti-Analysis Techniques

Latrodectus leverages several anti-analysis methods to hinder reverse-engineering efforts and detection by security solutions.

String Obfuscation

The malware makes heavy use of string obfuscation to hide artifacts from analysts. *Latrodectus* 1.1 utilized a pseudorandom number generator (PRNG), using a hardcoded seed, to derive the XOR key for deobfuscation. Obfuscated strings within *Latrodectus* 1.2 and 1.3 are stored using a relatively simple structure and deobfuscated when required. The strings are stored using the following binary format in Figure 2.

```
struct latrodectus_string {
        _DWORD dwKey;
        _WORD wSeed;
        _UCHAR uBuf[];
};
```

Figure 2 - Obfuscated string binary format

The obfuscation algorithm is simple, and makes use of XOR and ADD operations. A reimplemented version in Python of the deobfuscation algorithm can be observed below in Figure 3. It is noteworthy that the actual size of

the obfuscated string is derived from the XOR operation of the `wSeed` and `dwKey` fields.

```
def latro_deobf(buffer: bytes) -> str:
   string = cparser._latrodectus_string(buffer)

   size = (string.wSeed ^ string.dwKey) & 0xFFF
   out_buf = bytearray(size)

   for i in range(size):
       string.dwKey += 1
       out_buf[i] = (string.uBuf[i] + 10) & 0xFF
       out_buf[i] = (string.dwKey ^ string.uBuf[i]) & 0xFF

   return out_buf.decode("utf-8")
```

Figure 3 - String deobfuscation routine reimplementation

Dynamic API Resolution via Windows API Hashing

Latrodectus uses CRC32 to resolve Windows APIs at runtime by walking the `InMemoryOrderModuleList` found within the PEB to first retrieve the addresses of loaded `ntdll.dll` (`0x26797E77`) and `kernel32.dll` (`0x2ECA438C`) modules.

The structure and control-flow used by *Latrodectus* is reminiscent of the BlackLotus' API resolution routine, which is avaliable in open-source.

```
 struct latrodectus_api_entry
{
         _DWORD    dwFuncHash;
         _HMODULE* hModule;
         _LPVOID*  pFunc;
};
```

Figure 4 - API Table Entry

The API table is built sequentially during the initialisation phase of the malware. The building of the API table in C-pseudocode can be seen below in Figure 5.

```
 api_tbl[0].FunctionHash = 0xE0762FEB; /* NtAllocateVirtualMemory */
api_tbl[0].Module = &pNtdll;
api_tbl[0].Function = &NtAllocateVirtualMemory;
api_tbl[1].FunctionHash = 0xB46508B5; /* RtlGetVersion */
api_tbl[1].Module = &pNtdll;
api_tbl[1].Function = &RtlGetVersion;
```

Figure 5 - Building of Windows API function table based on CRC32 hash

Anti-Debug

Latrodectus makes use of the well-known and documented method of checking if the `IsDebugged` flag is set within the processes' PEB [6]. This is a common and effective method regularly employed to evade debuggers. A reimplementation in C-pseudocode can be observed in Figure 6. This is also likely borrowed code from the BlackLotus open-source project. [2]

```
 int64 latro::is_debugged()
{
  return util::current_peb()->IsDebugged;
}
```

Figure 6 - PEB based anti-debugging measure employed by *Latrodectus*

Host Environment Process Count Checks

The host environment checks within *Latrodectus* includes checking the number of processes which are running, with different thresholds per Windows NT version. It is likely these checks are used to evade sandbox, emulation-based approaches, or other analysis environments in which the number of processes would be irregular.

If the number of processes and version constraint matches, as defined in Table 2, the malware will simply exit. This anti-analysis technique is not unique to *Latrodectus*, and has been used by the likes of the EvilBunny implant.

Windows Version Constraint	Process Number Trigger
>= Windows 10	Less than 75
=> Windows Server 2003 R2 && <= Windows Server 2012/R2	Less than 50

Table 2 - Process check constraints per version

NTFS Visibility Obscured via Alternate Data Stream (ADS)

Latrodectus makes use of a trick to delete itself while the process is still running, making use of an alternative data stream (ADS) and a specific chain of API calls. The following sequence of events occurs to achieve this:

1. The path to the current running process is gathered using `GetModuleFileName`. A `HANDLE` to the file is then acquired via `CreateFile` with `DELETE` access.
2. Call `SetFileInformationByHandle` with the `FileRenameInfo` class `FileName` member set to `:wtfbbq`.
3. Again, call `SetFileInformationByHandle`, however with the `FileDispositionInfo` class member `DeleteFile` set to `TRUE`
4. Close the `HANDLE` to trigger the `DeleteFile`
5. The `HANDLE` is duplicated, then renamed to an ADS - in this instance, `:wtfbbq`.

This code has likely been borrowed from an open-source project [7] and slightly modified to use Latrodectus' API resolution and string obfuscation routine. The ADS name (`:wtfbbq`) remains unchanged. This is also used by RaspberryRobin, HelloXD Ransomware, DarkPower Ransomware and implemented in the Offensive Nim project.

Conclusion

Latrodectus is a now-prominent malware loader that leverages a variety of anti-analysis techniques to avoid detection and thwart reverse engineering efforts. Its reliance on open-source projects like BlackLotus, combined with the ability to quickly adapt and incorporate such code, demonstrates the adversary's commitment to leveraging publicly available resources to enhance their malware capabilities.

The adversary behind *Latrodectus* has operational resilience in the face of takedowns, demonstrating the adversary's commitment to maintaining a robust malware delivery infrastructure.

RevEng.AI Platform

RevEng.AI cuts down on the reverse-engineering time in the analysis stage by using our state of the art binary AI model. Using our AI Binary Analysis platform, analysts were quickly able to identify the differences between each *Latrodectus* sample based at a function-level and binary-level to provide an overview of differences implemented by the malware developer version-to-version. Alongside this, the overlaps with BlackLotus.

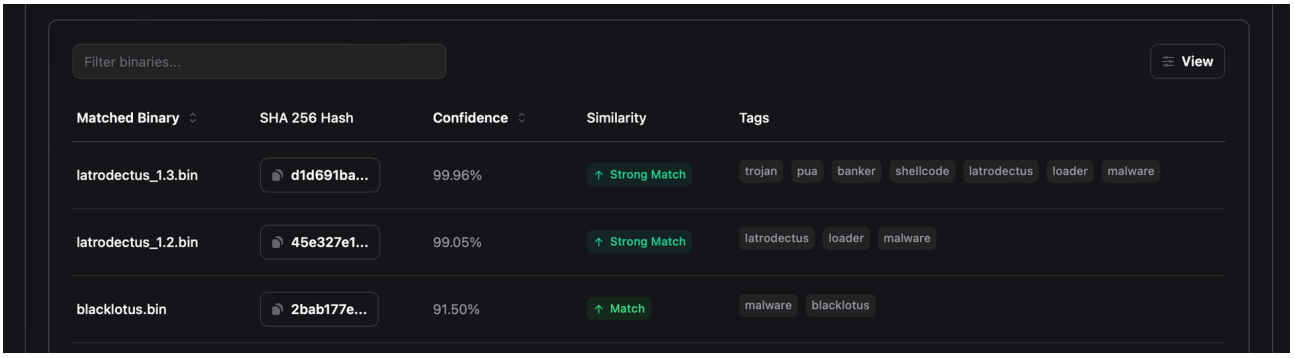

Figure 7 - RevEng.AI-identified Latrodectus and BlackLotus similarity analysis

RevEng.AI allows analysts to quickly and easily cluster malware samples based on the code similarity observed between binaries.

IOCs (Indicators of Compromise)

Host IOCs

SHA-256	Filename	Description
3ac8decd825d1ed7a86ed86d7789b44c0f0c467d4f482ab0863df1b7b1e3e8cc	Form_Ver-11-58-52.js	Malicious JavaScript Downloader
4586250dbf8cbe579662d3492dd33fe0b3493323d4a060a0d391f20ecb28abf1	vpn.msi	MSI file

SHA-256	Filename	Description
c89d15789fd9e0b23e62bbf038d2ddbcea5618573517f3382790b4b0434933df	disk1.cab	CAB archive containing packed BRC4 Badger
c5dc5fd676ab5b877bc86f88485c29d9f74933f8e98a33bddc29f0f3acc5a5b9	aclui.dll	Packed Brute Ratel Badger
0d3fd08d237f2f22574edf6baf572fa3b15281170f4d14f98433ddeda9f1c5b2	N/A, in-memory	Unpacked Brute Ratel Badger Stager
dbd85d5dd501bb7fad3990f0801d32da438a5bc60bd7cf6999d5bc535291146c	N/A, injected into explorer.exe	*Latrodectus* 1.3
44ccc3fbd3e15e8bdb063616d9baa37b1f9ab9121759fd467c943b7611860f72	N/A, injected into explorer.exe	*Latrodectus* Stealer Module

Table 3 - Host IOCs

Network IOCs

Host	Description
https[:]//lettecoft[.]com/live/	*Latrodectus* C2 Endpoint
https[:]//ultroawest[.]com/live/	*Latrodectus* C2 Endpoint
https[:]//kalopvard[.]com/live/	*Latrodectus* C2 Endpoint
https[:]//filomeranta[.]com/live/	*Latrodectus* C2 Endpoint
ultroawest[.]com	*Latrodectus* C2 Host
lettecoft[.]com	*Latrodectus* C2 Host
kalopvard[.]com	*Latrodectus* C2 Host
filomeranta[.]com	*Latrodectus* C2 Host

Host	Description
185.93.221[.]108	*Latrodectus* C2 IP
81.99.162[.]48	*Latrodectus* C2 IP
barsen[.]monster:7444	*Brute Ratel* C2
kurvabbr[.]pw:7444	*Brute Ratel* C2
94.232.249[.]86	*Brute Ratel* C2 IP
http[:]//85[.]208.108[.]63/BST.msi	MSI used by JavaScript Dropper

Table 4 - Network IOCs

Latrodectus Configuration

Key	Value
C2	https[:]//ultroawest[.]com/live/ https[:]//lettecoft[.]com/live/ https[:]//kalopvard[.]com/live/
RC4 Key	qNfSHTVKEU7mknHSFrQCwp0mmQfXUNPIcA66gezNz49qQOVX0P
Group	Mercury (0x88e6542e)
Version	1.3

Table 5 - *Latrodectus* Configuration

Footnotes

[1] - https://medium.com/walmartglobaltech/icedid-gets-loaded-af073b7b6d39, https://www.proofpoint.com/uk/blog/threat-insight/latrodectus-spider-bytes-ice
[2] - https://github.com/ldpreload/BlackLotus https://decoded.avast.io/janvojtesek/raspberry-robins-roshtyak-a-little-lesson-in-trickery/
[3] - https://learn.microsoft.com/en-us/windows/win32/msi/installer-installproduct
[4] - https://news.sophos.com/en-us/2023/05/18/the-phantom-menace-brute-ratel-remains-rare-and-targeted/
[5] - https://www.operation-endgame.com/
[6] - https://github.com/ldpreload/BlackLotus/blob/main/src/Bot/antidebug.c
[7] - https://github.com/LloydLabs/delete-self-poc/tags
