New HijackLoader Evasion Tactics | ThreatLabz

zscaler.com/blogs/security-research/analyzing-new-hijackloader-evasion-tactics

Zscaler Blog

Get the latest Zscaler blog updates in your inbox

Subscribe

Security Research

image

Introduction

HijackLoader (also known as IDAT Loader and GHOSTPULSE) is a malware loader initially discovered in 2023. The loader is not only capable of delivering second-stage payloads, but also offers a variety of modules to expand the malware's capabilities. The modules are mainly used for configuration information and to evade security software, as well as inject and execute code. Recently, Zscaler ThreatLabz uncovered new HijackLoader modules with additional evasion techniques. In this blog, we analyze these modules that implement features including call stack spoofing to mask the origin of function calls from endpoint detection, virtual machine detection to identify analysis environments, and another module that establishes persistence via scheduled tasks.

Key Takeaways

- HijackLoader is malware downloader dating back to 2023 that has received continuous updates via new modules.
- HijackLoader released a new module that implements call stack spoofing to hide the origin of function calls (e.g., API and system calls).
- HijackLoader added a new module to perform anti-VM checks to detect malware analysis environments and sandboxes.
- Another new module is designed to establish persistence via scheduled tasks.

Technical Analysis

In the following sections, we examine HijackLoader's new modules and changes in evasion tactics. The module names added since our last analysis are the following: ANTIVM, MUTEX, CUSTOMINJECT, CUSTOMINJECTPATH, modTask, modTask64, PERSDATA, and SM.

First stage

HijackLoader's first stage has undergone two changes. The first change involves the blocklist processes check, where a new process name, avastsvc.exe, was added to the list. If any of the processes in the table below are running, HijackLoader delays execution by 5 seconds.

SDBM Hash Value	Process Name	Description
5C7024B2	avgsvc.exe	The avgsvc.exe process is a component of AVG Internet Security.
6CEA4537	avastsvc.exe	The avastsvc.exe process is a component of Avast Antivirus.

Table 1: Processes blocklisted by HijackLoader.

The second change pertains to the decryption of modules. While most HijackLoader samples still use IDAT headers in a PNG file to store encrypted modules, a few samples are embedding them in the PNG's pixel structure.

Second stage (ti module)

As mentioned in our previous blog, HijackLoader uses the Heaven's Gate technique to execute x64 direct syscalls. We have now observed that call stack spoofing has been added to the list of evasion tactics used by HijackLoader. This technique uses a chain of EBP pointers to traverse the stack and conceal the presence of a malicious call in the stack by replacing actual stack frames with fabricated ones.

The ti module only uses call stack spoofing for the following native system APIs:

- ZwCreateSection
- ZwMapViewOfSection
- ZwUnmapViewOfSection
- ZwProtectVirtualMemory
- ZwReadVirtualMemoy
- ZwWriteVirtualMemory

- `ZwWriteFile`
- `ZwResumeThread`
- `ZwGetContextThread`
- `ZwSetContextThread`
- `ZwRollbackTransaction`
- `ZwClose`
- `ZwTerminateProcess`

HijackLoader API calls

HijackLoader collects the API hash, system call number, function name, and function address of all API names that start with *Zw* in `ntdll.dll`. This information is stored as an array of elements, with each element being a structure of size 16, which we will refer to as a `DIRECTSYSCALL_STRUCT`, as shown below.

```
struct DIRECTSYSCALL_STRUCT {
 uint32_t APIHash; // CRC32 hash of the API function name
 uint32_t ssn; // System service number (SSN)
 char *APIName; // API function name
 void *APIFunctionAddress; // API function address
};
```

When HijackLoader calls a Windows API function, the malware first locates the corresponding structure (`DIRECTSYSCALL_STRUCT`) for the specified API. HijackLoader then invokes the Windows API function either by directly calling its address (if not running under WOW64) or by utilizing a combination of call stack spoofing, Heaven's Gate, and direct syscalls (if running under WOW64).

Call stack spoofing

Call stack spoofing is used to mask the origin of function calls such as API and system calls. The figure below shows a high-level view of how HijackLoader leverages call stack spoofing:

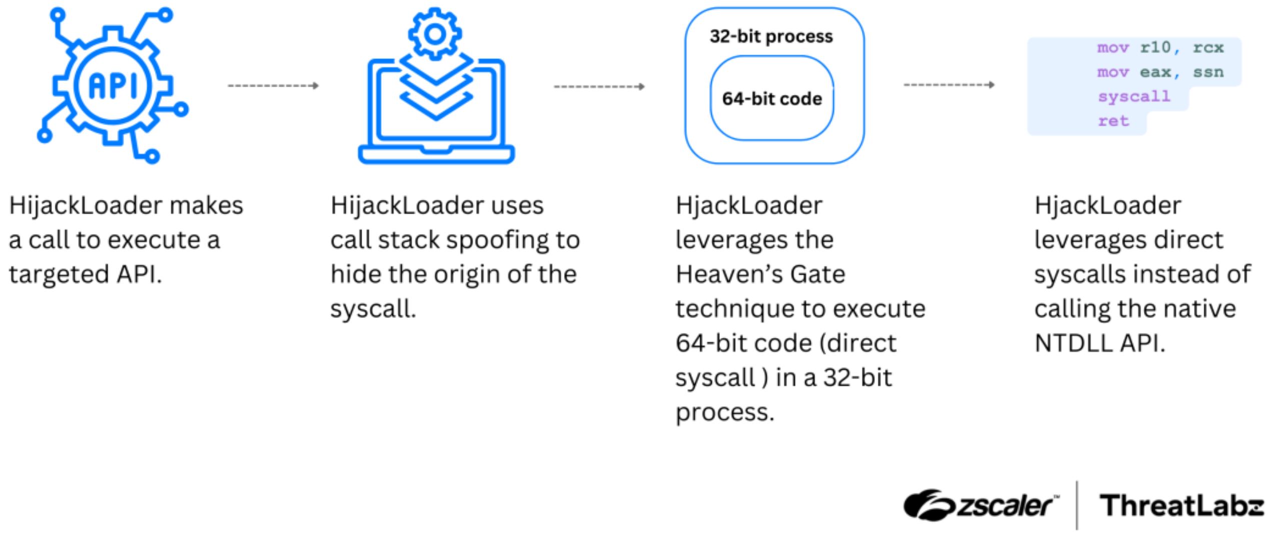


Figure 1: Diagram showing how HijackLoader uses call stack spoofing to mask the origin of function calls.

HijackLoader uses the base pointer register (EBP) to navigate the stack by following the chain of EBP pointers. The malware retrieves the return address pointer (EBP+4) from the stack frames. If the return address is not located in the text section of NTDLL or kernelbase, HijackLoader collects both the return address pointer and the return address of the stack frame. The return address pointer is then patched with a random address from the text section of a legitimate system DLL. This activity is repeated until the stack limit is reached or when three adjacent stack frames have the return address in the text section of NTDLL or kernelbase. The process is illustrated in the diagram below:

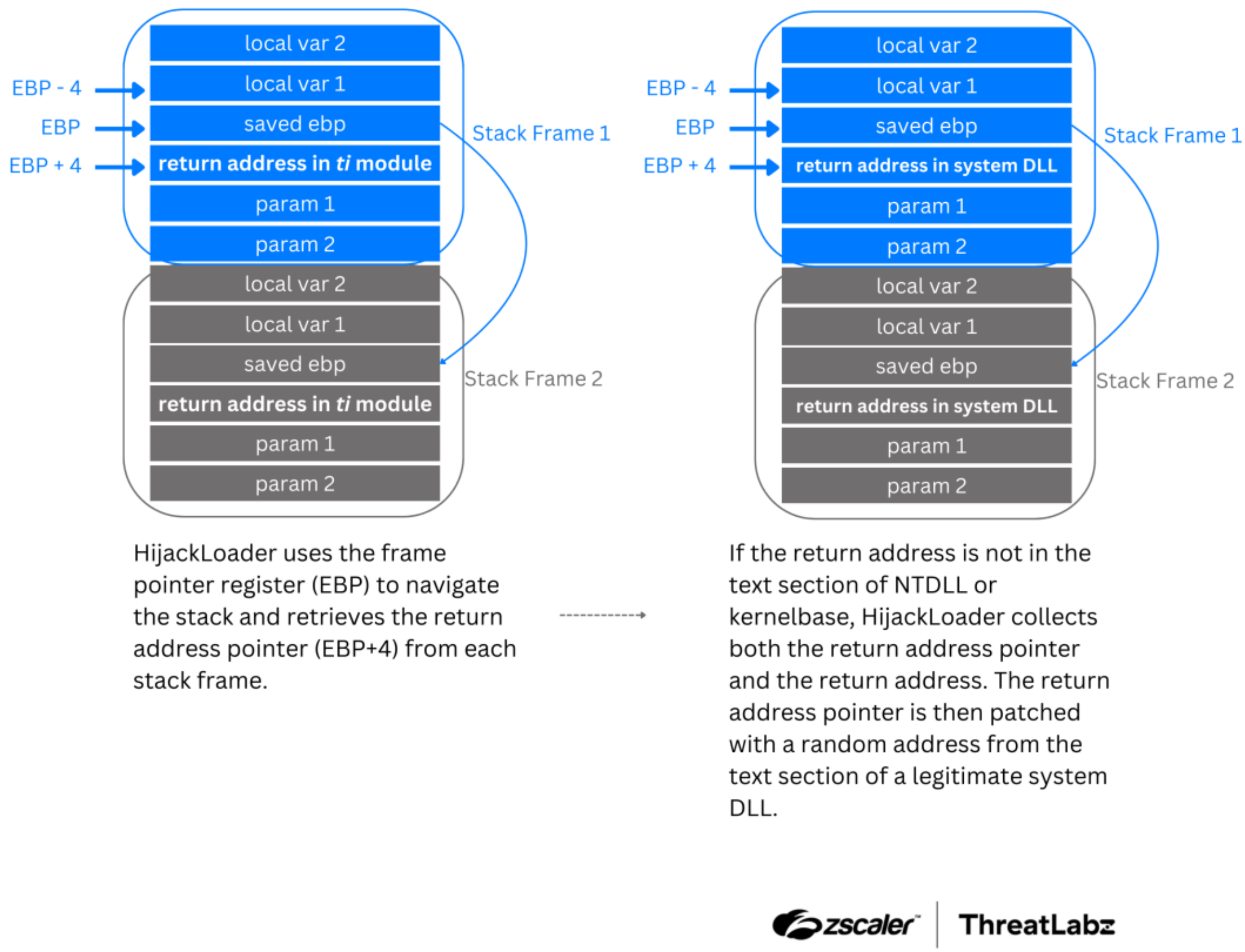


Figure 2: Diagram depicting how HijackLoader traverses the stack to retrieve and patch the return addresses to spoof stack frames.

The name of this legitimate system DLL is specified in the HijackLoader `SM` module. After this, HijackLoader employs the Heaven's Gate technique, which allows it to switch from executing 32-bit (x86) code to executing 64-bit (x64) code. Once in 64-bit mode, HijackLoader performs the direct syscall. HijackLoader uses the syscall number (`ssn`) and the necessary parameters for the native system service API to execute the direct syscall. Following the syscall, HijackLoader transitions back to x86 and patches the return address pointers with the actual return addresses.

Previously, HijackLoader utilized direct syscalls for process injection and to remap the `.text` section of the x64 `ntdll.dll` in memory with the `.text` section of the x64 `ntdll.dll` from disk. In addition to remapping `ntdll.dll`, HijackLoader now also remaps the `.text` section of the x64 `wow64cpu.dll` from disk to memory to remove user-mode hooks.

Other than the `ti` module, the modules `modCreateProcess`, `modUAC`, and `modTask` also use call stack spoofing. However, these modules do not use Heaven's Gate or make direct syscalls, but instead invoke Windows API functions directly.

The figure below shows the call stack for a call to `CreateProcessW` after the return addresses have been patched by the `modCreateProcess` module. In the call stack, the return addresses outside of the text section of NTDLL and kernelbase are patched with addresses from the text section of a legitimate system DLL (`shdocvw.dll`) until three adjacent stack frames have the return address in the text section of NTDLL or kernelbase.

Frame	Module	Location	Address
K 0	ntoskrnl.exe	NtFindAtom + 0x3ef	0xfffff8037fc423af
K 1	ntoskrnl.exe	PsWow64GetProcessMachine + 0xb22	0xfffff8037fc3c302
K 2	ntoskrnl.exe	FsRtlAllocateExtraCreateParameterList + 0x10d0	0xfffff8037fc72440
K 3	ntoskrnl.exe	setjmpex + 0x7c95	0xfffff8037fa0a9b5
U 4	ntdll.dll	NtCreateUserProcess + 0x14	0x7ffdcd20e8f4
U 5	wow64.dll	Wow64AllocThreadHeap + 0x13b9	0x7ffdcbb91739
U 6	wow64.dll	Wow64AllocThreadHeap + 0xb90	0x7ffdcbb90f10
U 7	wow64.dll	Wow64SystemServiceEx + 0x15a	0x7ffdcbb8901a
U 8	wow64cpu.dll	TurboDispatchJumpAddressEnd + 0xb	0x77ae17c3
U 9	wow64cpu.dll	BTCpuSimulate + 0x9	0x77ae11b9
U 10	wow64.dll	Wow64KiUserCallbackDispatcher + 0x4b9	0x7ffdcbb838c9
U 11	wow64.dll	Wow64LdrpInitialize + 0x12d	0x7ffdcbb832bd
U 12	ntdll.dll	LdrInitShimEngineDynamic + 0x33f7	0x7ffdcd2439c7
U 13	ntdll.dll	LdrInitializeThunk + 0x1db	0x7ffdcd1e4d2b
U 14	ntdll.dll	LdrInitializeThunk + 0x63	0x7ffdcd1e4bb3
U 15	ntdll.dll	LdrInitializeThunk + 0xe	0x7ffdcd1e4b5e
U 16	ntdll.dll	NtCreateUserProcess + 0xc	0x77b6371c
U 17	KernelBase.dll	CreateProcessInternalW + 0xce4	0x779c01f4
U 18	KernelBase.dll	CreateProcessW + 0x2c	0x779bf4fc
U 19	shdocvw.dll	OpenURL + 0x3	0x73992aab
U 20	shdocvw.dll	Ordinal142 + 0x19	0x73992a6b
U 21	shdocvw.dll	Ordinal221 + 0xb	0x73992bf5
U 22	shdocvw.dll	DllCanUnloadNow + 0xe6	0x73983806
U 23	shdocvw.dll	Ordinal185 + 0x7	0x73992b49
U 24	shdocvw.dll	Ordinal173 + 0x8	0x73992e30
U 25	shdocvw.dll	Ordinal123 + 0x5	0x73992e8f
U 26	shdocvw.dll	Ordinal145 + 0x7	0x73992c29
U 27	shdocvw.dll	SafeOpenPromptForShellExec + 0x98	0x7398ced8
U 28	shdocvw.dll	SetQueryNetSessionCount + 0x291	0x7398d201
U 29	shdocvw.dll	Ordinal138 + 0x6	0x73992eba
U 30	ntdll.dll	RtlIpv6AddressToStringA + 0x1c6	0x77b62a56
U 31	ntdll.dll	RtlActivateActivationContextUnsafeFast + 0xe2	0x77b3de02
U 32	ntdll.dll	RtlEqualUnicodeString + 0x5a3	0x77b41903
U 33	ntdll.dll	RtlEqualUnicodeString + 0x711	0x77b41a71
U 34	ntdll.dll	RtlIsCriticalSectionLockedByThread + 0xb5	0x77b42315
U 35	ntdll.dll	LdrLoadDll + 0x4b2	0x77b3e332
U 36	ntdll.dll	LdrLoadDll + 0xf6	0x77b3df76
U 37	KernelBase.dll	LoadLibraryExW + 0x156	0x779d1d96

U	38	KernelBase.dll	LoadLibraryA + 0x42	0x779d28b2
U	39	pyexec.exe	pyexec.exe + 0x1cf4	0x401cf4
U	40	kernel32.dll	BaseThreadInitThunk + 0x19	0x76e8fa29
U	41	ntdll.dll	RtlGetAppContainerNamedObjectPath + 0x11e	0x77b57b5e
U	42	ntdll.dll	RtlGetAppContainerNamedObjectPath + 0xee	0x77b57b2e

Figure 3: Example of HijackLoader spoofing the call stack with the fake frames enclosed in a red square.

Recent HijackLoader modules

The table below lists information about the more recent HijackLoader modules.

CRC32	Module Name	Description
`0x4dad7707`	`ANTIVM`	Contains the configurations HijackLoader uses for anti-VM checks (explained in detail in the following section).
`0x1999709f`	`MUTEX`	Contains a mutex name. If a mutex with this name exists, HijackLoader will exit.
`0x6703f815`	`CUSTOMINJECT`	Contains a legitimate executable file which is used for injecting code into its process memory. The process is created in a custom path specified by the `CUSTOMINJECTPATH` module.
`0x192a4446`	`CUSTOMINJECTPATH`	Contains a file path used to create the legitimate file in the `CUSTOMINJECT` module.
`0x3115355e`	`modTask`	Creates a scheduled task for persistence (explained in detail in the next section).
`0x9bfaf2d3`	`modTask64`	A 64-bit version of the `modTask` module.
`0xa2e0ab5d`	`PERSDATA`	Contains the configuration used by the `modTask` module to create scheduled tasks.

CRC32	Module Name	Description
`0xd8222145`	`SM`	Contains the name of the system DLL used in call stack spoofing to patch the return addresses. `TinycallProxy` module is also copied to this system DLL.
`0x455cbbc3`	`TinycallProxy`	Acts as a proxy to execute API calls. The call to the `TinycallProxy` module will have the address of the API function, number of parameters for the API call, and parameters for the API call as its arguments.
`0x5515dcea`	`TinycallProxy64`	This module is a 64-bit version of the `TinycallProxy` module.

Table 2: Description of more recent HijackLoader modules.

Virtual machine detection module

The virtual machine detection module `ANTIVM` contains a configuration used by HijackLoader to identify virtual machines and analysis environments. This configuration is stored in a structure which we will refer to as the `ANTIVM_STRUCT`, as shown below.

```
struct ANTIVM_STRUCT {
 uint32_t antiVMType;
 uint32_t timeThreshold;
 uint32_t minPhysicalMemory;
 uint32_t minProcessorCount;
 uint32_t antiVMType2;
 wchar_t username[20]; // Hardcoded to "george" (may change between samples)
 byte PhysicalMemory;
 byte ProcessorCount;
};
```

The first member, `antiVMType` determines the type of anti-VM check to be performed. These checks employ common anti-VM techniques. The `antiVMType` can include multiple values combined using bitwise OR operations. The values supported are listed in the table below.

Value	Check Performed
0x1	Calculates the average time taken to execute the *CPUID* instruction using the *RDTSC* instruction and compares it against the timeThreshold member of the `ANTIVM_STRUCT`. If the measured time equals or exceeds the timeThreshold, HijackLoader exits.

Value	Check Performed
0x4	Calls the *CPUID* instruction with EAX set to 1 and checks if the 31st bit of the ECX register (the hypervisor present bit) is set. If the bit is set, HijackLoader terminates.
0x8	Retrieves the maximum input value for hypervisor CPUID information by calling the *CPUID* instruction with EAX set to 0x40000000. If this value is greater than or equal to 0x40000000, HijackLoader exits. For instance, on Microsoft hypervisors, this value will be at least 0x40000005.
0x10	Retrieves the total physical memory of the system in gigabytes and compares it to the minPhysicalMemory member of the `ANTIVM_STRUCT`. If the total physical memory is less than or equal to minPhysicalMemory, HijackLoader exits.
0x20	Retrieves the number of processors on the system and compares it to the minProcessorCount member of the `ANTIVM_STRUCT`. If the processor count is less than or equal to minProcessorCount, HijackLoader exits.
0x40	Encompasses multiple checks, determined by the antiVMType2 member of the `ANTIVM_STRUCT.` The supported checks are: • 0x1 - Verifies if the computer name consists only of numbers. • 0x2 - Verifies if the username matches the username member of the `ANTIVM_STRUCT`. • 0x4 - Verifies if HijackLoader is executed from the Desktop folder or any of its subfolders. *ANALYST NOTE: These three checks appear to be in development, as HijackLoader does not exit even if the conditions are met.* Additionally, irrespective of the antiVMType2 value, HijackLoader compares the system's total physical memory in gigabytes with the PhysicalMemory member of the `ANTIVM_STRUCT` and the number of processors with the ProcessorCount member of the `ANTIVM_STRUCT`. If both of these checks are equal (which may be a specific configuration for a malware sandbox), HijackLoader exits.

Table 3: Description of values supported by the HijackLoader virtual machine detection module.

Persistence module

Before transferring control to the `modTask` persistence module, the `ti` module copies itself to a new address and the ti module copy is XOR'ed with the performance counter value obtained by calling the QueryPerformanceCounter API. The new address of the XOR'ed ti module and the XOR key are stored for restoration purposes.

When control is transferred to the `modTask` module, HijackLoader begins by overwriting the entire plaintext `ti` module with zeros. HijackLoader then performs call stack spoofing as previously described. Next, the `modTask` module copies the `TinycallProxy` module into the text section of the system DLL specified in the `SM` module and uses this copied `TinycallProxy` module to call APIs.

Then, HijackLoader creates a scheduled task for persistence using the configuration in the `PERSDATA` configuration module. The configuration is stored in a structure which we will refer to as the `PERSDATA_STRUCT`, with the definition shown below.

```
struct PERSDATA_STRUCT {
  uint32_t triggerTaskOnLogon; // If set, the task will be triggered when the
                               // user logs in, otherwise the task will execute
                               // at regular intervals.
  uint32_t TaskFlag;           // Flag used in ITask::SetFlags method
  uin32_t MinutesInterval;     // Task execution interval in minutes
  uin32_t wRandMinutesInterval;// Unused by the TASK_TRIGGER structure
  wchar_t taskName[50];        // Name of the task
};
```

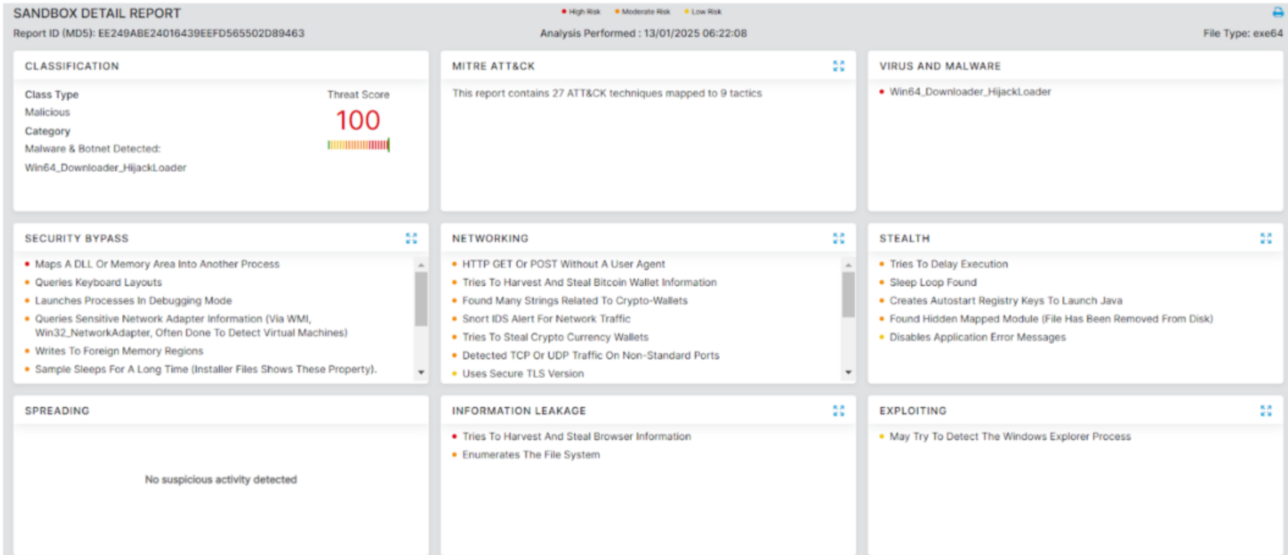
More information about HijackLoader's modules are available in our previous blog.

Conclusion

HijackLoader is a highly modular malware loader that shows no signs of slowing down. HijackLoader's new modules demonstrate the malware's evolving evasion tactics that increasingly focus on enhancing its anti-detection capabilities. Thus, we anticipate that HijackLoader will continue to introduce new modules that are further designed to complicate analysis and detection.

Zscaler Coverage

Zscaler's multilayered cloud security platform detects indicators related to HijackLoader at various levels. The figure below depicts the Zscaler Cloud Sandbox, showing detection details for HijackLoader.



zscaler | ThreatLabz

Figure 4: Zscaler Cloud Sandbox report for HijackLoader.

In addition to sandbox detections, Zscaler's multilayered cloud security platform detects indicators related to HijackLoader at various levels with the following threat names:

- Win64.Downloader.HijackLoader
- Win32.Downloader.HijackLoader

Indicators Of Compromise (IOCs)

SHA256 Hash	Description
67173036149718a3a06847d20d0f30616e5b9d6796e050dc520259a15588ddc8	HijackLoader sample
7b399ccced1048d15198aeb67d6bcc49ebd88c7ac484811a7000b9e79a5aac90	HijackLoader sample
6cfbffa4e0327969aeb955921333f5a635a9b2103e05989b80bb690f376e4404	HijackLoader sample
b2b5c6a6a3e050dfe2aa13db6f9b02ce578dd224926f270ea0a433195ac1ba26	HijackLoader sample
d75d545269b0393bed9fd28340ff42cc51d5a1bd7d5d43694dac28f6ca61df03	HijackLoader sample

SHA256 Hash	Description
9218c8607323d7667f69ef26faea57cb861f9b3888a457ed9093c1b65eefa42b	HijackLoader sample
b8f1341ade1fe50c4936b8f7bec7a8e47ad753465f716a1ec2f8220a18bf34a5	HijackLoader sample
35dca05612aede9c1db55a868b1cd314b5d05bac00bed577fd0d437103c2a4a4	HijackLoader sample
08f1ca6071cb206f53c2e81568b73d4bee7ac6a019d93d3ceaac7637b6dc891a	HijackLoader sample
b480fec95b84980e88e0e5958873b7194029ffbaa78369cfe5c0e4d64849fb32	HijackLoader sample
273bc7700e9153f7063b689f57ece3090c79e6b1038a9bc7865f61452c7377b0	HijackLoader encrypted modules.
28eb6ce005d34e22f6805a132e7080b96f236d627078bcc1bedee1a3a209bd1f	HijackLoader encrypted modules.
2be2c90c725c2a03d2bd68e39d52c0e16e7678d1d42fa7fdf75797806e0eb036	HijackLoader encrypted modules.
2e5cf739a84c726dfe3cfa3ddf47893357713240e77adf929ef30d87b1ccb52e	HijackLoader encrypted modules.
307c1756c21ee8f4f866ff8327823b55d597fecca379f98bcd45581e2e33adee	HijackLoader encrypted modules.
3142e4b40d27f63bcf7c787e96811e9a801224ce368624d75e88fa6408af896e	HijackLoader encrypted modules.

SHA256 Hash	Description
3500426eb9bb67fa91d4848cabeab2fe8e8a614768ed1e389e1f42a2428f64a8	HijackLoader encrypted modules.
3aa32545a2f53138d5f816d002b00d45c581cd56b1cfa66a2f72a03d604f1346	HijackLoader encrypted modules.
3ca78fbfbb46722af5f8acac511e77ec0382439f84c78c5710496fe1c377893d	HijackLoader encrypted modules.

MITRE ATT&CK Techniques

ID	Technique Name	Description
T1574.002	Hijack Execution Flow: DLL Side-Loading	HijackLoader samples mostly use DLL sideloading for execution.
T1027.007	Dynamic API Resolution	HijackLoader uses the SDBM hashing algorithm and CRC32 hashing algorithm for API resolution.
T1027.003	Steganography	HijackLoader uses steganography to hide its modules in a PNG image.
T1140	Deobfuscate/Decode Files or Information	HijackLoader uses XOR to decode its modules and final payload.
T1057	Process Discovery	HijackLoader checks for process names and compares them against antivirus security software.
T1620	Reflective Code Loading	The `ti` module is reflectively loaded by stomping it to a legitimate DLL using LoadLibrary and VirtualProtect.

ID	Technique Name	Description
T1547.001	Registry Run Keys / Startup Folder	HijackLoader creates a shortcut file (LNK) in the Windows Startup folder as one of its methods for persistence.
T1197	BITS Jobs	HijackLoader uses BITS Jobs to achieve persistence.
T1053	Scheduled Task/Job	HijackLoader's `modTask` module uses Windows Task Scheduler for persistence.
T1548.001	Abuse Elevation Control Mechanism	HijackLoader's `modUAC` modules use CMSTPLUA COM interface for UAC bypass.
T1055	Process Injection	HijackLoader uses process injection techniques to inject its final payload.
T1497	Virtualization/Sandbox Evasion	HijackLoader's `ANTIVM` modules contain multiple virtualization evasion techniques.
T1562.001	Impair Defenses: Disable or Modify Tools	HijackLoader's `WDDATA` module contains the PowerShell (PS) command for Windows Defender exclusion.

Thank you for reading

Was this post useful?

Yes, very!Not really

content is subject to change without notice. By accessing this blog, you agree to these terms and acknowledge your sole responsibility to verify and use the information as appropriate for your needs.

Explore more Zscaler blogs

A person typing at their computer screen where HijackLoader appears in large letters.
HijackLoader Updates

Read post

Modular cement blocks, side of building
Technical Analysis of HijackLoader

Read post

Pikabot reverse engineering
Technical Analysis of Pikabot

Read post

Get the latest Zscaler blog updates in your inbox

By submitting the form, you are agreeing to our privacy policy.