

Mem3nt0 mori – The Hacking Team is back!

Boris Larin : : 10/26/2025



Authors

-  [Boris Larin](#)

In March 2025, Kaspersky detected a wave of infections that occurred when users clicked on personalized phishing links sent via email. No further action was required to initiate the infection; simply visiting the malicious website using Google Chrome or another Chromium-based web browser was enough.

The malicious links were personalized and extremely short-lived to avoid detection. However, Kaspersky's technologies successfully identified a sophisticated zero-day exploit that was used to escape Google Chrome's sandbox. After conducting a quick analysis, we reported the vulnerability to the Google security team, who fixed it as [CVE-2025-2783](#).

[TBD][[405143032](#)] High CVE-2025-2783: Incorrect handle provided in unspecified circumstances in Mojo on Windows. Reported by Boris Larin (@oct0xor) and Igor Kuznetsov (@2igosha) of Kaspersky on 2025-03-20

Acknowledgement for finding CVE-2025-2783 (excerpt from the security fixes included into Chrome 134.0.6998.177/.178)

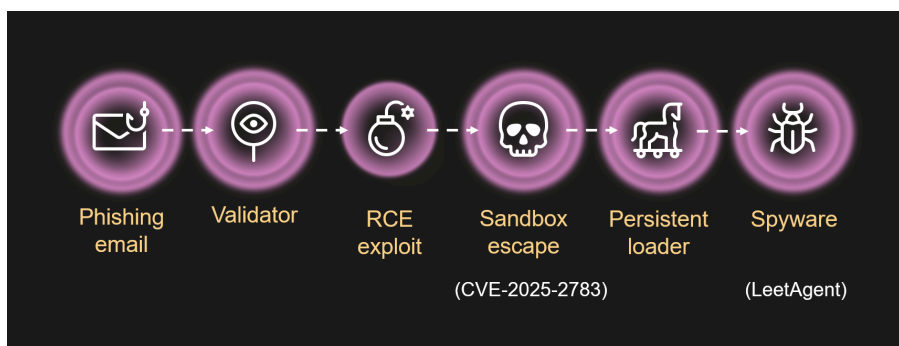
We dubbed this campaign [Operation ForumTroll](#) because the attackers sent personalized phishing emails inviting recipients to the Primakov Readings forum. The lures targeted media outlets, universities, research centers, government organizations, financial institutions, and other organizations in Russia. The functionality of the malware suggests that the operation's primary purpose was espionage.

We traced the malware used in this attack back to 2022 and discovered more attacks by this threat actor on organizations and individuals in Russia and Belarus. While analyzing the malware used in these attacks, we discovered an unknown piece of malware that we identified as commercial spyware called "Dante" and developed by the Italian company Memento Labs (formerly [Hacking Team](#)).

Similarities in the code suggest that the Operation ForumTroll campaign was also carried out using tools developed by Memento Labs.

In this blog post, we'll take a detailed look at the Operation ForumTroll attack chain and reveal how we discovered and identified the Dante spyware, which remained hidden for years after the Hacking Team rebrand.

Attack chain



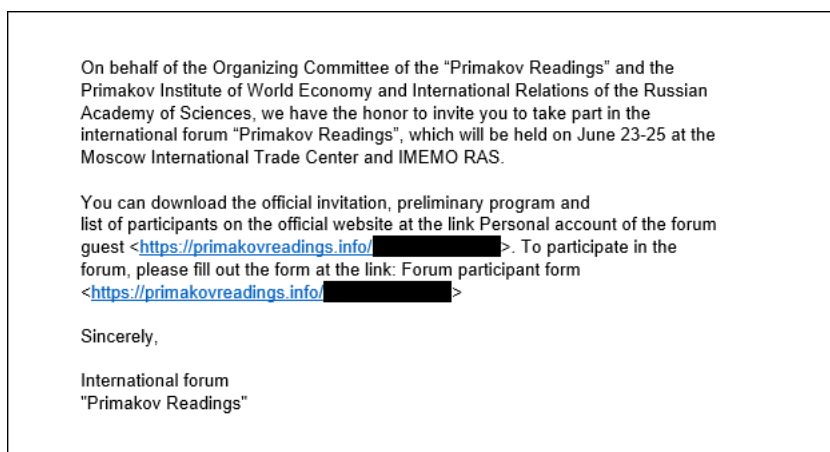
Operation ForumTroll attack chain

In all known cases, infection occurred after the victim clicked a link in a spear phishing email that directed them to a malicious website. The website verified the victim and executed the exploit.

When we first discovered and began analyzing this campaign, the malicious website no longer contained the code responsible for carrying out the infection; it simply redirected visitors to the official Primakov Readings website.

Therefore, we could only work with the attack artifacts discovered during the first wave of infections. Fortunately, Kaspersky technologies detected nearly all of the main stages of the attack, enabling us to reconstruct and analyze the Operation ForumTroll attack chain.

Phishing email



Example of a malicious email used in this campaign (translated from Russian)

The malicious emails sent by the attackers were disguised as invitations from the organizers of the Primakov Readings scientific and expert forum. These emails contained personalized links to track infections. The emails appeared authentic, contained no language errors, and were written in the style one would expect for an invitation to such an event. Proficiency in Russian and familiarity with local peculiarities are distinctive features of the ForumTroll APT group, traits that we have also observed in its other campaigns. However, mistakes in some of those other cases suggest that the attackers were not native Russian speakers.

Validator




The validator is a relatively small script executed by the browser. It validates the victim and securely downloads and executes the next stage of the attack.

The first action the validator performs is to calculate the SHA-256 of the random data received from the server using the WebGPU API. It then verifies the resulting hash. This is done using the open-source code of Marco Ciaramella's [sha256-gpu](#) project. The main purpose of this check is likely to verify that the site is being visited by a real user with a real web browser, and not by a mail server that might follow a link, emulate a script, and download an exploit. Another possible reason for this check could be that the exploit triggers a vulnerability in the WebGPU API or relies on it for exploitation.

The validator sends the infection identifier, the result of the WebGPU API check and the newly generated public key to the C2 server for key exchange using the Elliptic-curve Diffie-Hellman (ECDH) algorithm. If the check is passed, the server responds with an AES-GCM key. This key is used to decrypt the next stage, which is hidden in requests to

bootstrap.bundle.min.js and .woff2 font files. Following the timeline of events and the infection logic, this next stage should have been a remote code execution (RCE) exploit for Google Chrome, but it was not obtained during the attack.

Sandbox escape exploit

In-the-wild 0-days caught and reported by Kaspersky				
 Adobe	 Microsoft	Google		
CVE-2014-0497	CVE-2014-4077	CVE-2019-0859	CVE-2019-13720	CVE-2023-32434
CVE-2014-0515	CVE-2015-2360	CVE-2019-1458	CVE-2024-4947	CVE-2023-32435
CVE-2014-0546	CVE-2016-0034	CVE-2020-0986	CVE-2025-2783	CVE-2023-38606
CVE-2016-4171	CVE-2016-0165	CVE-2020-1380		CVE-2023-41990
CVE-2017-11292	CVE-2016-3393	CVE-2021-28310		
	CVE-2018-8174	CVE-2021-31955		
	CVE-2018-8453	CVE-2021-31956		
	CVE-2018-8589	CVE-2021-40449		
	CVE-2018-8611	CVE-2023-28252		
	CVE-2019-0797	CVE-2024-30051		

List of in-the-wild 0-days caught and reported by Kaspersky

Over the years, we have discovered and reported on dozens of zero-day exploits that were actively used in attacks. However, CVE-2025-2783 is one of the most intriguing sandbox escape exploits we've encountered. This exploit genuinely puzzled us because it allowed attackers to bypass Google Chrome's sandbox protection without performing any obviously malicious or prohibited actions. This was due to a powerful logical vulnerability caused by an obscure quirk in the Windows OS.

To protect against bugs and crashes, and enable sandboxing, Chrome [uses](#) a multi-process architecture. The main process, known as the browser process, handles the user interface and manages and supervises other processes. [Sandboxed](#) renderer processes handle web content and have limited access to system resources. Chrome uses [Mojo](#) and the underlying [ipc](#) library, introduced to replace legacy IPC mechanisms, for interprocess communication between the browser and renderer processes.

The exploit we discovered came with its own Mojo and ipc libraries that were statically compiled from official sources. This enabled attackers to communicate with the IPC broker within the browser process without having to manually craft and parse ipc messages. However, this created a problem for us because, to analyze the exploit, we had to identify all the Chrome library functions it used. This involved a fair amount of work, but once completed, we knew all the actions performed by the exploit.

In short, the exploit does the following:

- Resolves the addresses of the necessary functions and code gadgets from dll using a pattern search.
- Hooks the `v8_inspector::V8Console::Debug` function. This allows attackers to escape the sandbox and execute the desired payload via a JavaScript call.
- Starts executing a sandbox escape when attackers call `console.debug(0x42, shellcode);` from their script.
- Hooks the `ipc::NodeLink::OnAcceptRelayedMessage` function.
- Creates and sends an ipc message of the type `RelayMessage`. This message type is used to pass Windows OS handles between two processes that do not have the necessary permissions (e.g., renderer processes). The exploit retrieves the handle returned by the [GetCurrentThread](#) API function and uses this ipc message to relay it to itself. The broker transfers handles between processes using the [DuplicateHandle](#) API function.
- Receives the relayed message back using the `ipc::NodeLink::OnAcceptRelayedMessage` function hook, but instead of the handle that was previously returned by the [GetCurrentThread](#) API function, it now contains a handle to the thread in the browser process!
- Uses this handle to execute a series of code gadgets in the target process by suspending the thread, setting register values using `SetThreadContext`, and resuming the thread. This results in shellcode execution in the browser process and subsequent installation of a malware loader.

So, what went wrong, and how was this possible? The answer can be found in the descriptions of the [GetCurrentThread](#) and [GetCurrentProcess](#) API functions. When these functions are called, they don't return actual handles; rather, they return *pseudo handles*, special constants that are interpreted by the kernel as a handle to the

current thread or process. For the current process, this constant is -1 (also equal to `INVALID_HANDLE_VALUE`, which brings its own set of quirks), and the constant for the current thread is -2. Chrome's IPC code already [checked](#) for handles equal to -1, but there were no checks for -2 or other undocumented pseudo handles. This oversight led to the vulnerability. As a result, when the broker passed the -2 pseudo handle received from the renderer to the `DuplicateHandle` API function while processing the `RelayMessage`, it converted -2 into a real handle to its own thread and passed it to the renderer.

Shortly after the patch was released, it became clear that Chrome was not the only browser affected by the issue. Firefox developers quickly identified a similar pattern in their IPC code and released an update under [CVE-2025-2857](#).

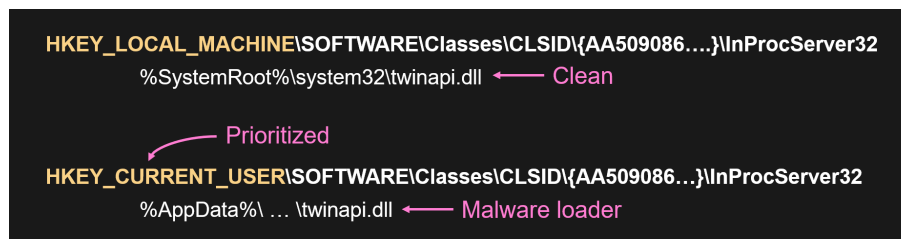
When pseudo handles were first introduced, they simplified development and helped squeeze out extra performance – something that was crucial on older PCs. Now, decades later, that outdated optimization has come back to bite us.

Could we see more bugs like this? Absolutely. In fact, this represents a whole class of vulnerabilities worth hunting for – [similar](#) issues may still be lurking in other applications and Windows system services.

To learn about the hardening introduced in Google Chrome following the discovery of CVE-2025-2783, we recommend checking out Alex Gough's [upcoming](#) presentation, "Responding to an ITW Chrome Sandbox Escape (Twice!)," at Kawaiiicon.

Persistent loader

Persistence is achieved using the Component Object Model (COM) hijacking technique. This method exploits a system's search order for COM objects. In Windows, each COM class has a registry entry that associates the CLSID (128-bit GUID) of the COM with the location of its DLL or EXE file. These entries are stored in the system registry hive `HKEY_LOCAL_MACHINE (HKLM)`, but can be overridden by entries in the user registry hive `HKEY_CURRENT_USER (HKCU)`. This enables attackers to override the CLSID entry and run malware when the system attempts to locate and run the correct COM component.



COM hijacking in a nutshell

The attackers used this technique to override the CLSID of `twinapi.dll` {AA509086-5Ca9-4C25-8F95-589D3C07B48A} and cause the system processes and web browsers to load the malicious DLL.

This malicious DLL is a loader that decrypts and executes the main malware. The payload responsible for loading the malware is encoded using a simple binary encoder similar to those found in the Metasploit framework. It is also obfuscated with OLLVM. Since the hijacked COM object can be loaded into many processes, the payload checks the name of the current process and only loads the malware when it is executed by certain processes (e.g., `rdpclip.exe`). The main malware is decrypted using a modified ChaCha20 algorithm. The loader also has the functionality to re-encrypt the malware using the BIOS UUID to bind it to the infected machine. The decrypted data contains the main malware and a shellcode generated by Donut that launches it.

LeetAgent

LeetAgent is the spyware used in the Operation ForumTroll campaign. We named it LeetAgent because all of its commands are written in [leetspeak](#). You might not believe it, but this is rare in APT malware. The malware connects to one of its C2 servers specified in the configuration and uses HTTPS to receive and execute commands identified by unique numeric values:

- 0xC033A4D (COMMAND) – Run command with `cmd.exe`
- 0xECEC (EXEC) – Execute process
- 0x6E17A585 (GETTASKS) – Get list of tasks that agent is currently executing
- 0x6177 (KILL) – Stop task
- 0xF17E09 (FILE \x09) – Write file
- 0xF17ED0 (FILE \xD0) – Read file
- 0x1213C7 (INJECT) – Inject shellcode
- 0xC04F (CONF) – Set communication parameters

- 0xD1E (DIE) – Quit
- 0xCD (CD) – Change current directory
- 0x108 (JOB) – Set parameters for keylogger or file stealer

In addition to executing commands received from its C2, it runs keylogging and file-stealing tasks in the background. By default, the file-stealer task searches for documents with the following extensions: *.doc, *.xls, *.ppt, *.rtf, *.pdf, *.docx, *.xlsx, *.pptx.

The configuration data is encoded using the TLV (tag-length-value) scheme and encrypted with a simple single-byte XOR cipher. The data contains settings for communicating with the C2, including many settings for traffic obfuscation.

In most of the observed cases, the attackers used the Fastly.net cloud infrastructure to host their C2. Attackers frequently use it to download and run additional tools such as 7z, Rclone, SharpChrome, etc., as well as additional malware (more on that below).

The number of traffic obfuscation settings may indicate that LeetAgent is a commercial tool, though we have only seen ForumTroll APT use it.

Finding Dante

In our opinion, attributing unknown malware is the most challenging aspect of security research. Why? Because it's not just about analyzing the malware or exploits used in a single attack; it's also about finding and analyzing all the malware and exploits used in past attacks that might be related to the one you're currently investigating. This involves searching for and investigating similar attacks using indicators of compromise (IOCs) and tactics, techniques, and procedures (TTPs), as well as identifying overlaps in infrastructure, code, etc. In short, it's about finding and piecing together every scrap of evidence until a picture of the attacker starts to emerge.

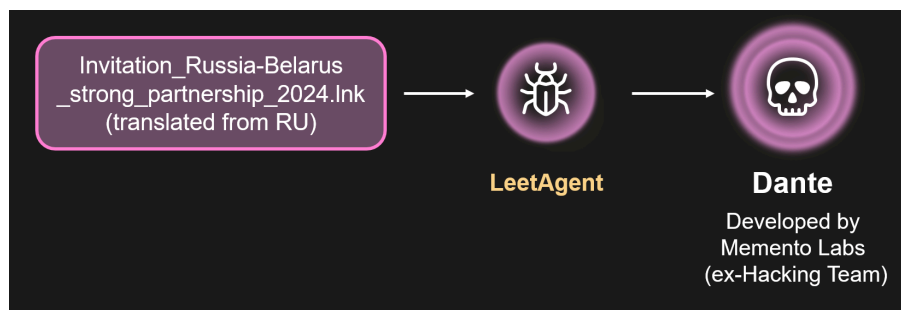
We traced the first use of LeetAgent back to 2022 and discovered more ForumTroll APT attacks on organizations and individuals in Russia and Belarus. In many cases, the infection began with a phishing email containing malicious attachments with the following names:

- Baltic_Vector_2023.iso (translated from Russian)
- DRIVE.GOOGLE.COM (executable file)
- Invitation_Russia-Belarus_strong_partnership_2024.Ink (translated from Russian)
- Various other file names mentioning individuals and companies

In addition, we discovered another cluster of similar attacks that used more sophisticated spyware instead of LeetAgent. We were also able to track the first use of this spyware back to 2022. In this cluster, the infections began with phishing emails containing malicious attachments with the following names:

- SCAN_XXXX_<DATE>.pdf.Ink
- <DATE>_winscan_to_pdf.pdf.Ink
- Rostelecom.pdf.Ink (translated from Russian)
- Various others

The attackers behind this activity used similar file system paths and the same persistence method as the LeetAgent cluster. This led us to suspect that the two clusters might be related, and we confirmed a direct link when we discovered attacks in which this much more sophisticated spyware was launched by LeetAgent.



Connection between LeetAgent and commercial spyware called Dante

After analyzing this previously unknown, sophisticated spyware, we were able to identify it as commercial spyware called Dante, developed by the Italian company Memento Labs.

The Atlantic Council's Cyber Statecraft Initiative recently [published](#) an interesting report titled "Mythical Beasts and where to find them: Mapping the global spyware market and its threats to national security and human rights." We

think that comparing commercial spyware to mythical beasts is a fitting analogy. While everyone in the industry knows that spyware vendors exist, their “products” are rarely discovered or identified. Meanwhile, the list of companies developing commercial spyware is huge. Some of the most famous are NSO Group, Intellexa, Paragon Solutions, Saito Tech (formerly Candiru), Vilicius Holding (formerly FinFisher), Quadream, Memento Labs (formerly Hacking Team), negg Group, and RCS Labs. Some are always in the headlines, some we have [reported](#) on before, and a few have almost completely faded from view. One company in the latter category is Memento Labs, formerly known as [Hacking Team](#).

Hacking Team (also stylized as HackingTeam) is one of the oldest and most famous spyware vendors. Founded in 2003, Hacking Team became known for its Remote Control Systems (RCS) spyware, used by government clients worldwide, and for the many controversies surrounding it. The company’s trajectory changed dramatically in 2015 when more than 400 GB of internal data was leaked online following a hack. In 2019, the company was acquired by InTheCyber Group and renamed Memento Labs. “We want to change absolutely everything,” the Memento Labs owner [told](#) Motherboard in 2019. “We’re starting from scratch.” Four years later, at the ISS World MEA 2023 [conference](#) for law enforcement and government intelligence agencies, Memento Labs revealed the name of its new surveillance tool – DANTE. Until now, little was known about this malware’s capabilities, and its use in attacks had not been discovered.

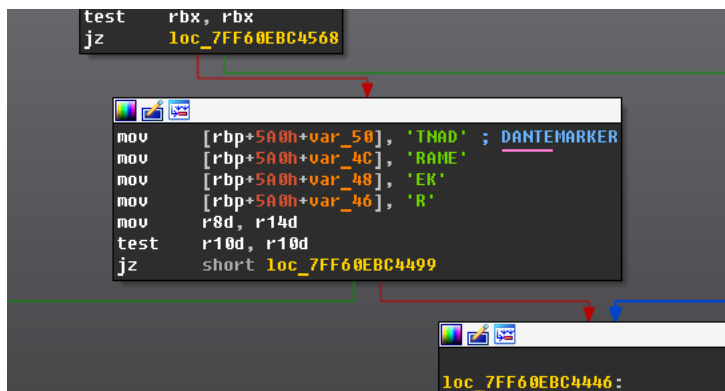
16:15-17:00 Session C

Set and forget: automatizing the deep monitoring process usinf DANTE 2.0

Presented by Memento Labs

Excerpt from the agenda of the ISS World MEA 2023 conference (the typo was introduced on the conference website)

The problem with detecting and attributing commercial spyware is that vendors typically don’t include their copyright information or product names in their exploits and malware. In the case of the Dante spyware, however, attribution was simple once we got rid of VMProtect’s obfuscation and found the malware name in the code.



Dante spyware name in the code

Dante

Of course, our attribution isn’t based solely on the string “Dante” found in the code, but it was an important clue that pointed us in the right direction. After some additional analysis, we found a reference to a “2.0” version of the malware, which matches the title of the aforementioned conference talk. We then searched for and identified the most recent samples of Hacking Team’s Remote Control Systems (RCS) spyware. Memento Labs kept improving its codebase until 2022, when it was replaced by Dante. Even with the introduction of the new malware, however, not everything was built from scratch; the later RCS samples share quite a few similarities with Dante. All these findings make us very confident in our attribution.

Why did the authors name it Dante? This may be a nod to tradition, as RCS spyware was also known as “Da Vinci”. But it could also be a reference to Dante’s poem [Divine Comedy](#), alluding to the many “circles of hell” that malware analysts must pass through when detecting and analyzing the spyware given its numerous anti-analysis techniques.

First of all, the spyware is packed with VMProtect. It obfuscates control flow, hides imported functions, and adds anti-debugging checks. On top of that, almost every string is encrypted.

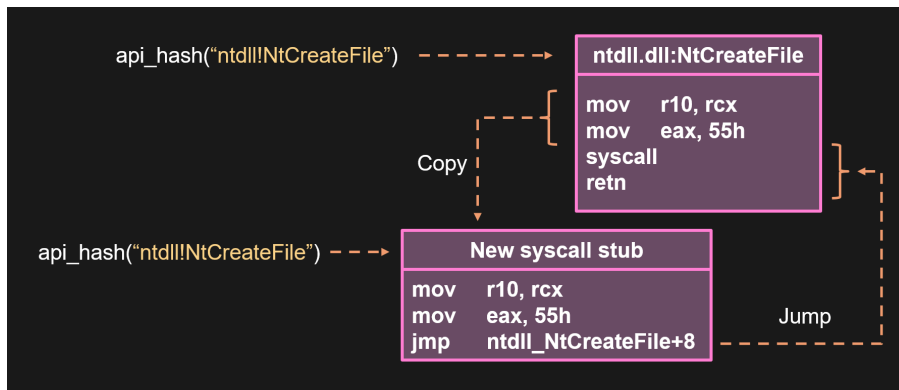

```

call    sub_7FF60ED14F52
mov     r14, rax Breakpoint here (e.g. step over) is skipped
mov     [rsp+100h+var_B8], rax Breakpoint here cause crash
test    rax, rax <----- Real return address
jz      short near ptr loc_7FF60EBE2767+1

```

VMProtect anti-debugging technique

To protect against dynamic analysis, Dante uses the following anti-hooking technique: when code needs to execute an API function, its address is resolved using a hash, its body is parsed to extract the system call number, and then a new system call stub is created and used.



Dante anti-hooking technique (simplified)

In addition to VMProtect's anti-debugging techniques, Dante uses some common methods to detect debuggers. Specifically, it checks the debug registers (Dr0–Dr7) using `NtGetContextThread`, inspects the `KdDebuggerEnabled` field in the `KUSER_SHARED_DATA` structure, and uses `NtQueryInformationProcess` to detect debugging by querying the `ProcessDebugFlags`, `ProcessDebugPort`, `ProcessDebugObjectHandle`, and `ProcessTlsInformation` classes.

To protect itself from being discovered, Dante employs an interesting method of checking the environment to determine if it is safe to continue working. It queries the [Windows Event Log](#) for events that may indicate the use of malware analysis tools or virtual machines (as a guest or host).

PROCMON2	VMWare	Oracle VM VirtualBox
ProcessHacker	pri_	\\Device\\VBoxNet
processhacker.sys	NECVMWAr	VMware NAT Service
ida64.exe	_VBOX&	VMware Player
hex-rays.com	\\MWVM	
rohitab.com	_FLOPPY_	
API Monitor	VBOX HARDDISK	
apimonitor-	VBoxService.exe	
API_Monitor_	VirtualBox Guest	
Wireshark	VMUpgradeHelper	
lnpcap.sys	VMTools	

The strings Dante searches for in the event logs

It also performs several anti-sandbox checks. It searches for “bad” libraries, measures the execution times of the `sleep()` function and the `cpuid` instruction, and checks the file system.

Some of these anti-analysis techniques may be a bit annoying, but none of them really work or can stop a professional malware analyst. We deal with these techniques on an almost daily basis.

After performing all the checks, Dante does the following: decrypts the configuration and the orchestrator, finds the string “DANTEMARKER” in the orchestrator, overwrites it with the configuration, and then loads the orchestrator.

The configuration is decrypted from the data section of the malware using a simple XOR cipher. The orchestrator is decrypted from the resource section and poses as a font file. Dante can also load and decrypt the orchestrator from the file system if a newer, updated version is available.

The orchestrator displays the code quality of a commercial product, but isn't particularly interesting. It is responsible for communication with C2 via HTTPs protocol, handling modules and configuration, self-protection, and self-removal.

Modules can be saved and loaded from the file system or loaded from memory. The infection identifier (GUID) is encoded in Base64. Parts of the resulting string are used to derive the path to a folder containing modules and the path to additional settings stored in the registry.

```
Infection identifier: 641d7fca-41a5-42fb-9bd3-4992147bd47a
Base64 encoded identifier: NjQxZDdmY2EtNDZhNS00MmZiLTliZDMtNDk5MjE0N2JkNDdh
Modules folder: %LocalAppData%\NjQxZDdm
Registry path: HKCU\Software\Classes\.zdmtnD\OpenWithProgids
```

An example of Dante's paths derivation

The folder containing modules includes a binary file that stores information about all downloaded modules, including their versions and filenames. This metadata file is encrypted with a simple XOR cipher, while the modules are encrypted with AES-256-CBC, using the first 0x10 bytes of the module file as the IV and the key bound to the machine. The key is equal to the SHA-256 hash of a buffer containing the CPU identifier and the Windows Product ID.

To protect itself, the orchestrator uses many of the same anti-analysis techniques, along with additional checks for specific process names and drivers.

If Dante doesn't receive commands within the number of days specified in the configuration, it deletes itself and all traces of its activity.

At the time of writing this report, we were unable to analyze additional modules because there are currently no active Dante infections among our users. However, we would gladly analyze them if they become available. Now that information about this spyware has been made public and its developer has been identified, we hope it won't be long before additional modules are discovered and examined. To support this effort, we are sharing a method that can be used to identify active Dante spyware infections (see the [Indicators of compromise](#) section).

Although we didn't see the ForumTroll APT group using Dante in the Operation ForumTroll campaign, we have observed its use in other attacks linked to this group. Notably, we saw several minor similarities between this attack and others involving Dante, such as similar file system paths, the same persistence mechanism, data hidden in font files, and other minor details. Most importantly, we found similar code shared by the exploit, loader, and Dante. Taken together, these findings allow us to conclude that the Operation ForumTroll campaign was also carried out using the same toolset that comes with the Dante spyware.

Conclusion

This time, we have not one, but three conclusions.

- 1) DuplicateHandle is a dangerous API function. If the process is privileged and the user can provide a handle to it, the code should return an error when a pseudo-handle is supplied.
- 2) Attribution is the most challenging part of malware analysis and threat intelligence, but also the most rewarding when all the pieces of the puzzle fit together perfectly. If you ever dreamed of being a detective as a child and solving mysteries like Sherlock Holmes, Miss Marple, Columbo, or Scooby-Doo and the Mystery Inc. gang, then threat intelligence might be the right job for you!
- 3) Back in 2019, Hacking Team's new owner stated in an interview that they wanted to change everything and start from scratch. It took some time, but by 2022, almost everything from Hacking Team had been redone. Now that Dante has been discovered, perhaps it's time to start over again.

Full details of this research, as well as future updates on ForumTroll APT and Dante, are available to customers of the APT reporting service through our Threat Intelligence Portal.

Contact: intelreports@kaspersky.com

Indicators of compromise

Kaspersky detections

Exploit.Win32.Generic
Exploit.Win64.Agent
Trojan.Win64.Agent

Trojan.Win64.Convagent.gen
HEUR:Trojan.Script.Generic
PDM:Exploit.Win32.Generic
PDM:Trojan.Win32.Generic
UDS:DangerousObject.Multi.Generic

Folder with modules

The folder containing the modules is located in %LocalAppData%, and is named with an eight-byte Base64 string. It contains files without extensions whose names are also Base64 strings that are eight bytes long. One of the files has the same name as the folder. This information can be used to identify an active infection.

Loader

7d3a30dbf4fd3edaf4dde35ccb5cf926
3650c1ac97bd5674e1e3bfa9b26008644edacfed
2e39800df1cafbefba22b437744d80f1b38111b471fa3eb42f2214a5ac7e1f13

LeetAgent

33bb0678af6011481845d7ce9643cedc
8390e2ebdd0db5d1a950b2c9984a5f429805d48c
388a8af43039f5f16a0673a6e342fa6ae2402e63ba7569d20d9ba4894dc0ba59

Dante

35869e8760928407d2789c7f115b7f83
c25275228c6da54cf578fa72c9f49697e5309694
07d272b607f082305ce7b1987bfa17dc967ab45c8cd89699bcdced34ea94e126

- [Malware Technologies](#)
- [Vulnerabilities and exploits](#)
- [Targeted attacks](#)
- [Google Chrome](#)
- [Malware Descriptions](#)
- [Spyware](#)
- [Cyber espionage](#)
- [Malware](#)
- [APT](#)
- [Hacking Team](#)
- [ForumTroll](#)
- [Dante](#)

Authors

-  [Boris Larin](#)

Mem3nt0 mori – The Hacking Team is back!

Your email address will not be published. Required fields are marked *

[Cancel](#)

This site uses Akismet to reduce spam. [Learn how your comment data is processed.](#)