ThreatMon

Technical Analysis of RDPCredentialStealer: Uncovering Malware Targeting RDP Credentials with API Hooking

@ThreatMon
@MonThreat
@threatmon
@TMRansomMonitor

Contents

Introduction

The rapid growth of remote work and the increased reliance on remote desktop protocols (RDP) have created new avenues for cybercriminals to exploit vulnerabilities in order to gain unauthorized access to sensitive information. One such threat is the RDPCredentialStealer, a malicious software designed to surreptitiously extract credentials entered by users during RDP sessions. This report provides a comprehensive technical analysis of the RDPCredentialStealer malware, detailing its functionality, attack vectors, and potential impact.

The RDPCredentialStealer employs a sophisticated technique called API Hooking with Detours, implemented using C++ programming language. By intercepting and redirecting application programming interface (API) calls, the malware covertly captures and exfiltrates sensitive login information provided by unsuspecting RDP users. Through an in-depth examination of its code and behavior, this report sheds light on the inner workings of the RDPCredentialStealer, enabling security professionals to better understand the threat landscape and develop effective countermeasures.

To aid in the detection and prevention of RDPCredentialStealer attacks, this report also presents a comprehensive set of defensive measures. It includes a YARA rule, a powerful tool for identifying patterns and characteristics of malware, specifically tailored to detect instances of the RDPCredentialStealer. Additionally, the report outlines relevant Mitre Att&ck techniques utilized by the malware, providing insights into the broader tactics, techniques, and procedures employed by threat actors.

Furthermore, this report shares important indicators of compromise (IOCs) associated with the RDPCredentialStealer, allowing security analysts to proactively identify and respond to potential infections.

By delving into the technical aspects of the RDPCredentialStealer, this report aims to enhance the understanding of this particular threat and empower organizations and cybersecurity professionals with the knowledge and tools necessary to safeguard their networks and data from this insidious form of attack.

Initial Analysis

Attack Chain

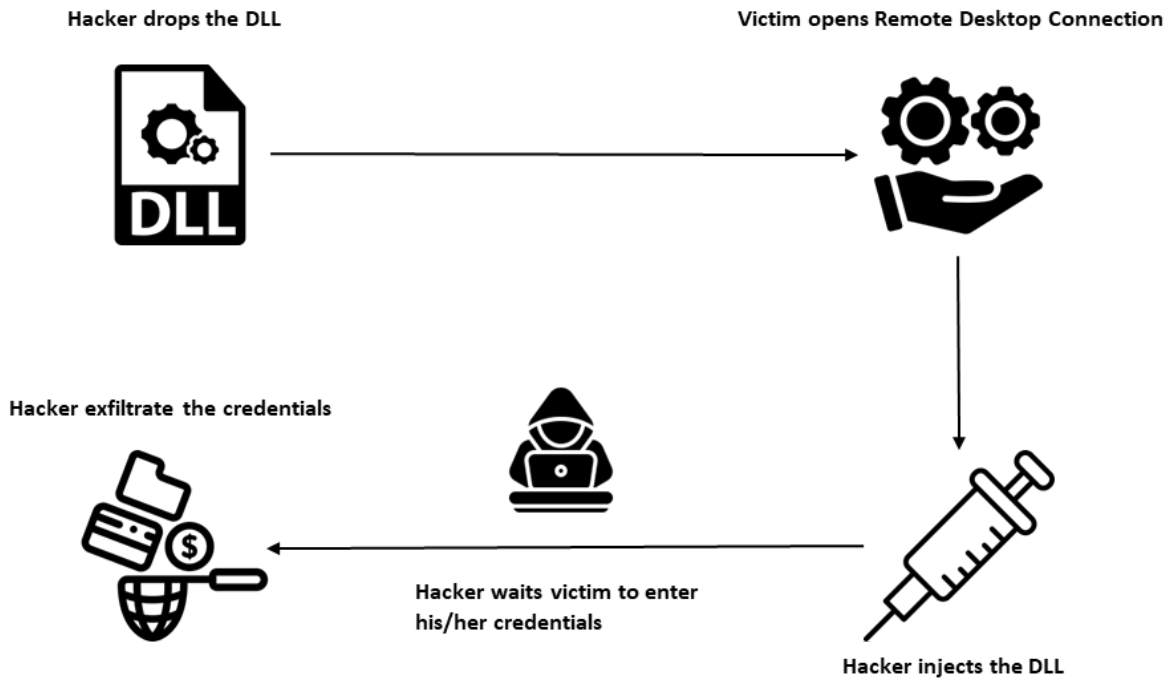

Figure 1 - Attack Chain Illustration

Dynamic Analysis

RDPCredentialStealer is a malware that steals credentials provided by users in RDP. It uses API Hooking technique. It consists of two parts: a DLL and an injector executable.

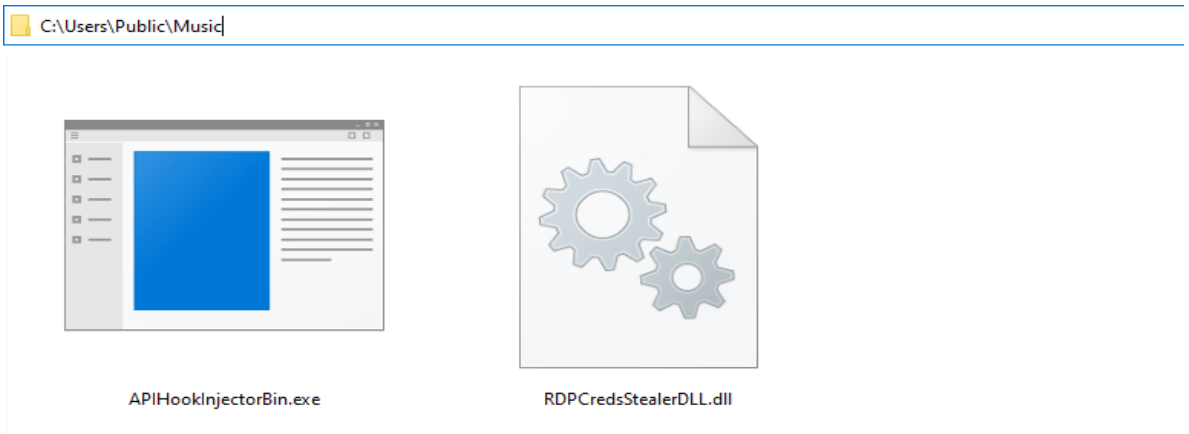

Figure 2 - Dropping the DLL under C:\Users\Public\Music

Initially, we place the DLL file in the directory C:\Users\Public\Music, following which we proceed to launch the RDP Connector and subsequently run the injector executable.

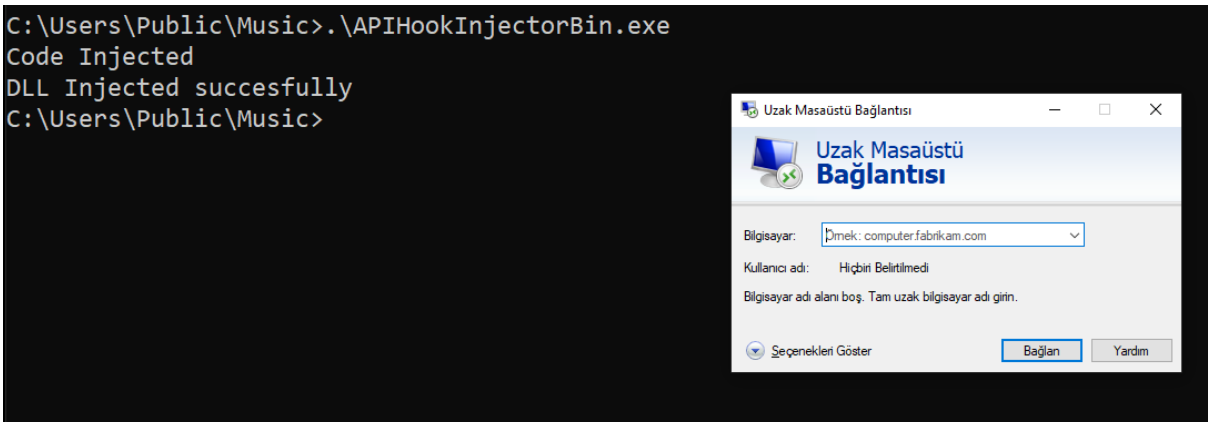

Figure 3 - Injecting the DLL

Following that, we input our RDP credentials.

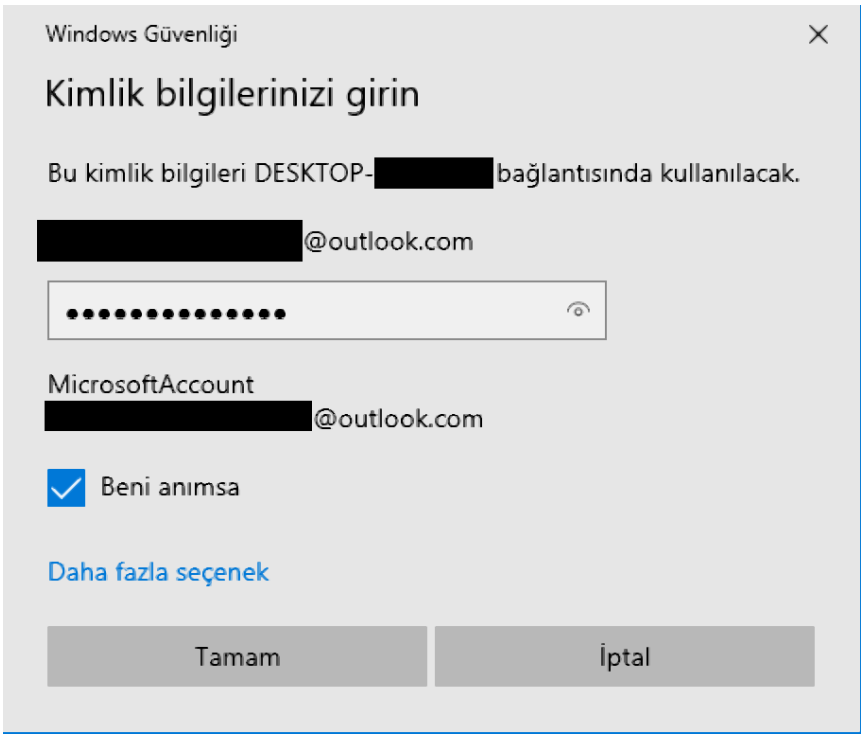

Figure 4 - Entering the Credentials

Afterward, a file named RDPCreds.txt will be generated in the directory C:\Users\Public\Music, which will include our credentials.

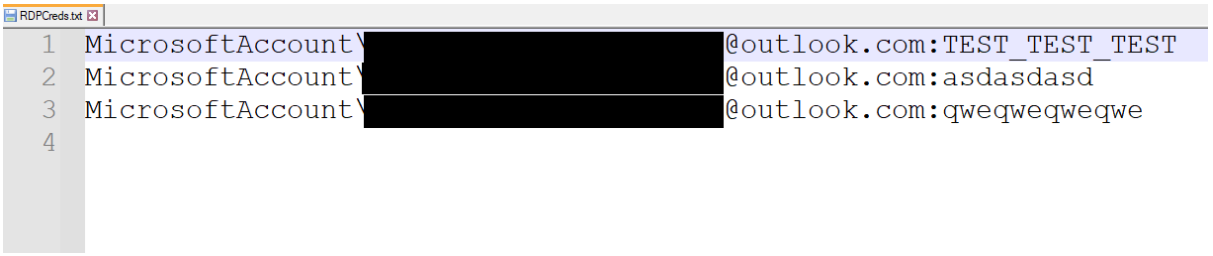

Figure 5 - Writes credentials under C:\Users\Public\Music\RDPCreds.txt

Hook Injector Executable

Name	APIHookInjectorBin
MD5	f9c0c006400a0be00c4cc7268b0f7c9a
SHA256	118c2a7d06f9ac1aabdec653f236e04f3a697f59bef6f4e9c9ca1ea8acdc33db
File Type	PE/64

Static Analysis

property	value
md5	F9C0C006400A0BE00C4CC7268B0F7C9A
sha1	9FA71973EA69C8340E5ACED5F254950FECDC00E2
sha256	118C2A7D06F9AC1AABDEC653F236E04F3A697F59BEF6F4E9C9CA1EA8ACDC33DB
first-bytes (hex)	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
first-bytes (text)	M Z .. .. @ .. ..
size	16384 bytes
entropy	5.178
imphash	191552F57F71544AD3A74066541944FF
cpu	64-bit
signature	n/a
entry-point (hex)	48 83 EC 28 E8 1B 04 00 00 48 83 C4 28 E9 72 FE FF
file-version	n/a
file-description	n/a
file-type	executable
subsystem	Console
compiler-stamp	Mon Jun 12 16:16:46 2023
debugger-stamp	Mon Jun 12 16:16:46 2023

Figure 6 - Overview to the malware

The first look at the malware shows that the compile and debug date is Jun 12, 2023.

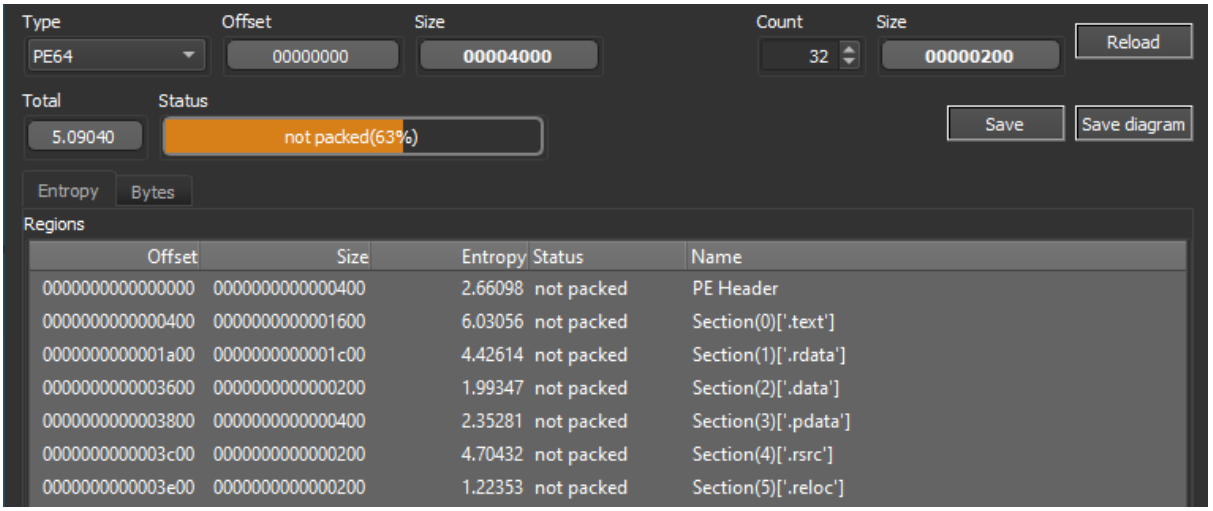

Figure 7 - Analysis of packing and code obfuscation

During the analysis, it is seen that there is no packaging process in the malware and there is no code obfuscation.

Dynamic Analysis

Figure 8 - Directory where the malware searches for the target dll

It is seen that the malware looks for the .dll file named RDPCredsStealerDLL, which it will use to perform the hook operation, under the "C:\\Users\\\Public\\\Music" directory.

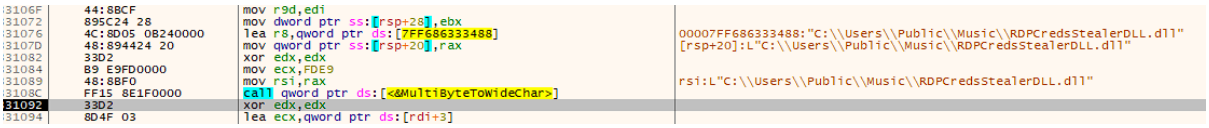

Figure 9 - Convert dll directory from ascii to unicode

The malware uses the **MultiByteToWideChar** API to convert the target DLL's index from ASCII characters to wide characters (Unicode). This is because the functions and APIs used in the Windows operating system support the wide character (Unicode) format. Therefore, when DLL injection is performed, the DLL file path must be in wide character format.

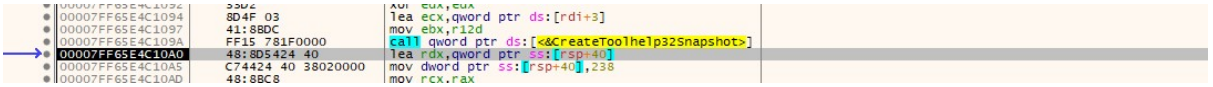

Figure 10 - Get a list of running processes with the CreateToolHelp32Snapshot API

Malware lists running processes using the **CreateToolHelp32Snapshot** API.

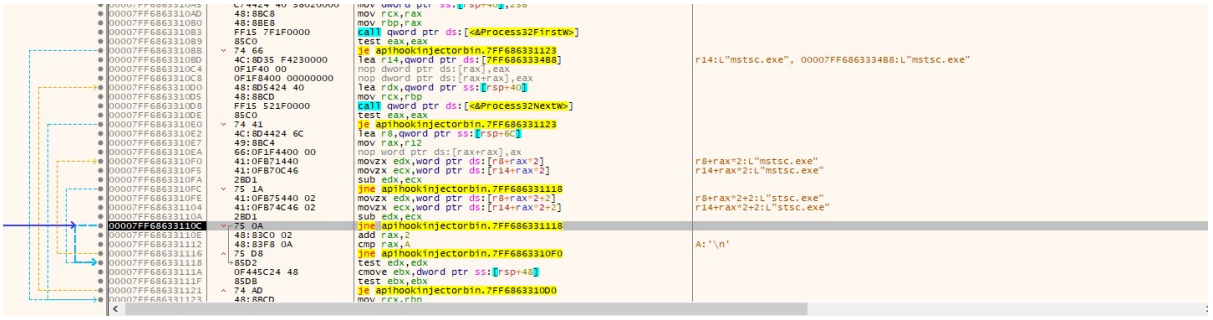

Figure 11 - Get a list of running processes with the CreateToolHelp32Snapshot API

It targeted the mstsc.exe used for Remote Desktop Connection to inject the malicious dll file. Afterwards, it returns the first process listed by the **CreateToolhelp32Snapshot** API using the **Process32FirstW** API. Then, within a loop, it uses the **Process32NextW** API to sequentially return all processes in the list until it matches the targeted **"mstsc.exe"**.

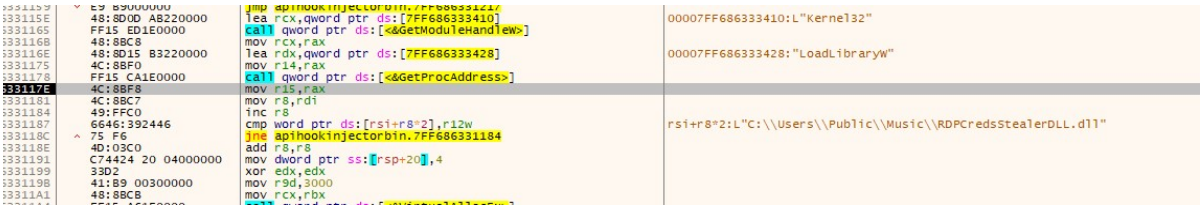

Figure 12 - DLL and API parsing

The malware executes **kernel32.dll** as a result of parsing the **GetModuleHandleW** API and then executes **LoadLibraryW** as a result of parsing the **GetProcAddress** API.

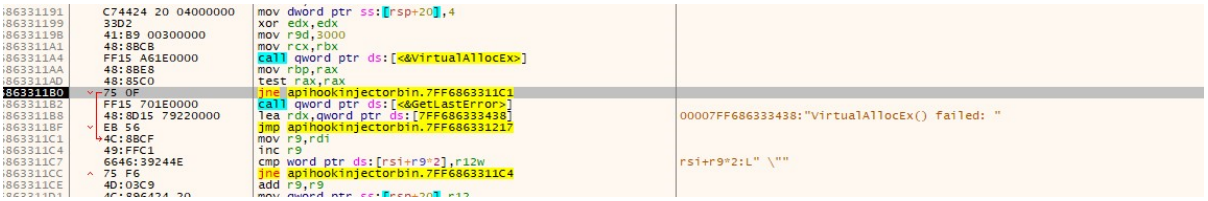

Figure 13 - Allocating mstsc.exe's memory

Malware allocates a space in the memory of mstsc.exe using the **VirtualAllocEx** API in order to inject RDPCredsStealerDLLdll.

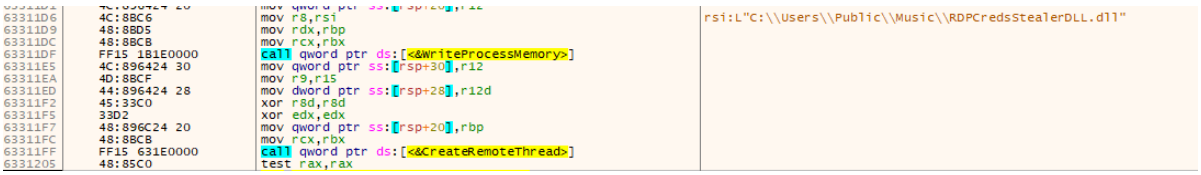

Figure 14 - Inject the malicious dll into the targeted process's memory

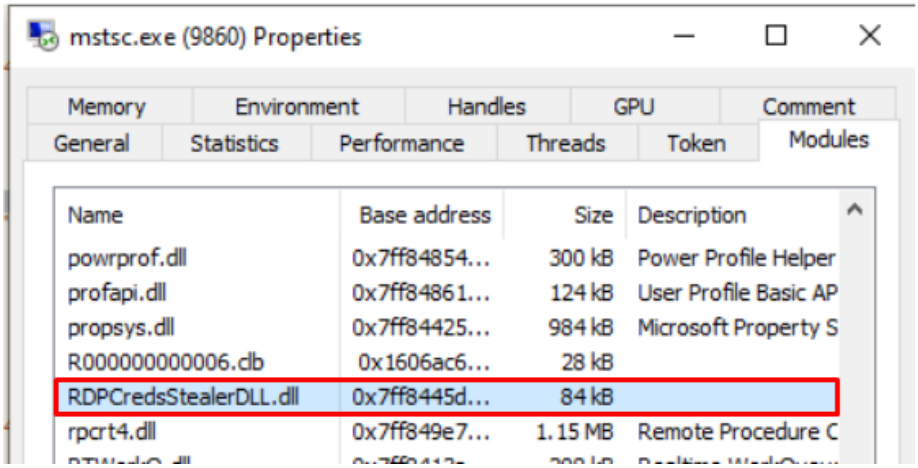

Figure 15 - Injected malicious dll file

It writes to the memory of mstsc.exe using the **WriteProcessMemory** API to inject the malicious dll into the memory of the malicious target process. Then malware runs the injected malicious dll file using the **CreateRemoteThread** API.

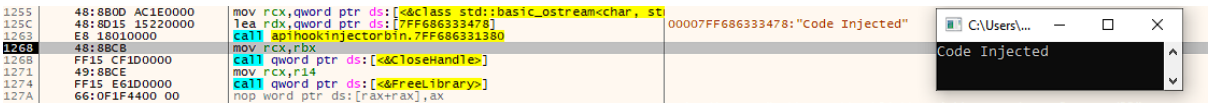

Figure 16 - Return message to command screen

The malware uses the **GetLastError** function to write the string "Code Injected" to the command screen it opens. Then it closes kernel32.dll using the **FreeLibrary** API.

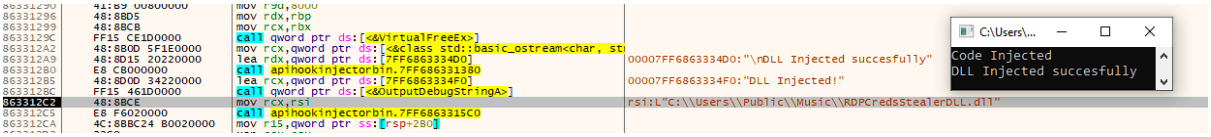

Figure 17 - Deleting traces of malicious file

The malicious code uses the **VirtualFreeEx** API to release the memory region allocated by **VirtualAllocEx** API in order to remove any traces of its presence in the memory of the target process. Then, using the **OutputDebugStringA** API, it writes the string "DLL Injected Successfully" to the command line and malware stops working.

Credential Stealer DLL

File Name	RDPCredsStealerDLL
MD5	8dfeb6f0ace5bed512c3dc12df1e27a7
SHA256	5eabd7d957e56a9cb9a918f7e9f72dc76a0481954c2f93ad5264095b5dbb6897
File Type	PE64/DLL

Static Analysis

property	value
md5	8DFEB6F0ACE5BED512C3DC12DF1E27A7
sha1	5399BC8FBBFE8E4E2F8945A43195A3EF0643CE65
sha256	5EABD7D957E56A9CB9A918F7E9F72DC76A0481954C2F93AD5264095B5DBB6897
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
first-bytes-text	M Z . . . . . @ . . . . .
file-size	63488 bytes
entropy	5.338
imphash	51BACF922C2C362C2D17A713EC0A53AB
signature	n/a
tooling	Visual Studio 2008
entry-point	48 89 5C 24 08 48 89 74 24 10 57 48 83 EC 20 49 8B F8 8B DA 48 8B F1 83 FA 01 75 05 E8 E3 05 00 00
file-version	n/a
description	n/a
file-type	dynamic-link-library
cpu	64-bit
subsystem	GUI
compiler-stamp	Mon Jun 12 23:06:51 2023 | UTC
debugger-stamp	Mon Jun 12 23:06:51 2023 | UTC

Figure 18 - Overview to the DLL

The first look at the malicious DLL shows that the compile and debug date is Jun 12, 2023.

Dynamic Analysis

Figure 19 - Dynamically Loads credui.dll

“credui.dll” performs tasks such as verifying credentials such as username and password or presenting an authentication screen to the user. It takes the user's credentials and forwards them to the right places.

The stealer loads the “credui.dll” file and applies a hook to the **CredUnPackAuthenticationBufferW** function to capture the password information entered by the user during the RDP connection.

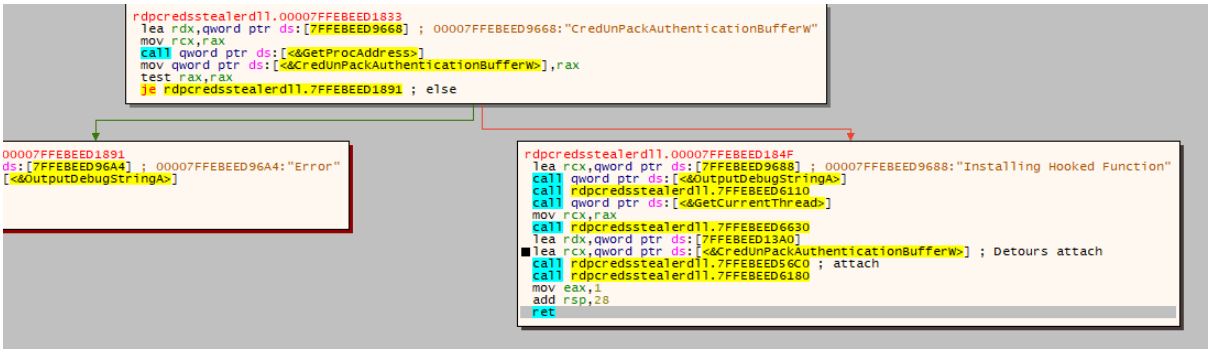

Figure 20 - API Hooking with Detours

The program uses the DetourTransactionBegin, DetourUpdateThread and DetourAttach functions to use its own API instead of the **CredUnPackAuthenticationBufferW** API.

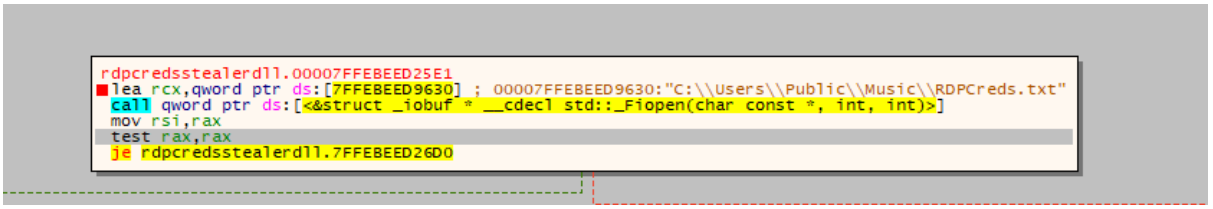

Figure 21 - Writes the credentials under C:\\Users\\Public\\Music

In this way, it creates a file named **RDPCreds.txt** in **C:\\Users\\Public\\Music** and saves the username and password written to RDP.

MITRE ATT&CK

Technique Name	Technique ID
Process Injection	T1055
Dynamic-Link Library Injection	T1055.001
Thread Execution Hijacking	T1055.003
Process Discovery	T1057
Input Capture: Credential API Hooking	T1056.004

Mitigations

- Some endpoint security solutions can be configured to block some types of process injection based on common sequences of behavior that occur during the injection process.
- Implement security controls on the endpoint, such as a Host Intrusion Prevention System (HIPS), to identify and prevent execution of files with mismatching file signatures.
- Utilize Yama (ex: /proc/sys/kernel/yama/ptrace_scope) to mitigate ptrace based process injection by restricting the use of ptrace to privileged users only. Other mitigation controls involve the deployment of security kernel modules that provide advanced access control and process restrictions such as SELinux, grsecurity, and AppArmor.
- Monitor for API calls may attempt to get information about running processes on a system.
- Monitor for newly executed processes that may attempt to get information about running processes on a system.
- Verify integrity of live processes by comparing code in memory to that of corresponding static binaries, specifically checking for jumps and other instructions that redirect code flow.

Detection

For YARA Rules and Indicators of Compromise (IOCs) check our github.

Get 30 Days Premium Access!

Get started with a limited, free version of Threatmon to analyze internet assets, operationalize attacker intelligence and transform your security program from reactive to proactive.

ThreatMon

45305 Catalina cs St 150, Sterling VA 20166