

Remote Clipboard Data In-Memory Attacks and Detection

Khaled Fawzy Mohamed, Nashwa AbdelBaki and Ahmed Shosha

Nile University, Cairo, Egypt

Abstract

The exchange of data using the clipboard feature between different applications and computers is a fundamental capability provided by modern operating systems and remote access tools. However, it can be abused by cybercriminals and attackers to obtain or manipulate sensitive data, such as passwords, secrets, and credential tokens. This research paper explores the various types of attacks that can be executed on the clipboard, with particular emphasis on new attacks on shared clipboard data. The investigation entails an in-depth examination of the interplay between the system clipboard and remote computers, including those accessed through Virtualization applications and Remote Desktop Protocol (RDP) sessions. The study also proposes a detection technique for identifying such attacks, which can be employed to distinguish between normal clipboard-sharing actions and malicious clipboard data sniffing and manipulation. Our findings demonstrate that clipboard hijacking can be accomplished remotely without the need to install malware on the victim's device. These findings underscore the significance of heightened awareness and security measures to thwart such attacks on shared clipboards. The attack can be successfully executed on all versions of Windows operating systems.

Keywords

attack, clipboard, data manipulation, detection, malware, RDP, remote access, virtualization

1. Introduction

Modern operating systems provide features to facilitate transferring data between applications, for example, System Clipboard [1]. However, cybercriminals and threat actors continue to develop new techniques to perform malicious activities, with the clipboard becoming a prime target for attacks. Malware can monitor and hijack the clipboard, allowing cybercriminals to steal sensitive data such as passwords and credit card numbers [2]. Clipboard hijacking involves taking control of the victim's clipboard, and replacing or removing its contents [3]. This attack has been performed on various operating systems, with Android applications in Google Play found to perform clipboard hijacking to steal seed phrases from the mobile's clipboard storage [4]. Additionally, cryptocurrency addresses have been targeted by cybercriminals using clipboard hijacking, resulting in 2.3 million addresses being monitored. Previous clipboard hijacking attacks required the installation of malware on the victim's device to control and manipulate the system clipboard [5]. Clipboard hijacking in previously mentioned attacks requires having the malware installed on the victim's device to be able to control the system clipboard and manipulate it [3].

The evolution of cybercriminals and threat actors has resulted in the emergence of new attacks, such as Remote Desktop Protocol (RDP) ransomware attacks. RDP is a popular protocol for remote administration and data transfer, connecting remote computers and clients on

both Windows and non-Windows machines, with clipboard data being one of the data types that can be transferred between RDP peers [6]. Virtualization applications like VMWare share the clipboard using the same technique as RDP, making it vulnerable to the newly discovered attack technique. Remote access applications like TeamViewer also share clipboard data using the same method, making them susceptible to this type of attack. However, VirtualBox shares clipboard data differently between guest and host operating systems.

Clipboard monitoring is a commonly used technique [7] to detect RDP ransomware attacks. Therefore, it is crucial to detect and prevent clipboard data manipulation attacks remotely through RDP, virtualization applications, and other remote access applications that allow clipboard-sharing. This research highlights that clipboard hijacking can occur remotely without the need for malware installation on a device. This study focused on the offensive utilization of this technique and found that the attack works on all major versions of Windows operating system. Thus, any windows-host connecting to a malicious Windows server infected with malware using this technique is susceptible to clipboard data manipulation. Additionally, our research team developed a real-time detection technique to distinguish between normal clipboard-sharing operations and malicious clipboard data manipulation.

2. Background

A clipboard is a set of messages and functions provided by modern operating systems to transfer various types of data between executing applications on the system [8]. Copy and Paste operations are used to transfer data from

✉ ka.fawzy@nu.edu.eg (K. F. Mohamed); nabelbaki@nu.edu.eg (N. AbdelBaki); ashosha@nu.edu.eg (A. Shosha)

© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).


 CEUR Workshop Proceedings (CEUR-WS.org)

Table 1
Clipboard data formats [9].

Constant	Value	Description
CF_TEXT	1	Null-terminated text format.
CF_OEMTEXT	7	Null-terminated text format containing characters in the OEM character set.
CF_UNICODETEXT	13	Null-terminated unicode text format.
N/A	49171	Private data formats, formats that are application specific.

the clipboard to different applications [1, 8]. The system clipboard supports multiple data formats[9], such as Files and Text within both Unicode and ASCII formats, and assigns a unique numeric format ID and textual name to each clipboard format to allow the destination application to identify and extract the data in the right way without extracting the data and parsing it later [10]. Essential clipboard operations include placing extra data from the clipboard, enumerating the available data formats on the clipboard, and registering itself to receive notifications in case of clipboard updates [11]. In the realm of Microsoft Windows, two primary clipboard mechanisms exist: The Standard Windows Clipboard API and Object Linking and Embedding (OLE) uniform data transfer (UDT) [11]. While Microsoft recommends the use of the OLE mechanism for certain use cases, the standard Clipboard Windows API is still supported and will continue to be maintained according to Microsoft’s confirmation [12]. To ensure the synchronization of independent system clipboards, monitoring clipboard updates is a technique that involves either polling the contents of the clipboard at regular intervals or registering for clipboard update notifications. Additionally, delayed rendering is a supportive technique that assists in keeping independent system clipboards in sync with minimal cost. This technique involves transferring only the format ID of the targeted data to be copied to the clipboard, as opposed to the actual data, which is then transferred upon request during the paste operation [13].

This study focused on text clipboard data formats as shown in Table 1.

3. Clipboard Data Sharing Analysis

The Remote Desktop Protocol (RDP) is a communication protocol that enables the communication between local and remote clipboards via the clipboard virtual channel extension. This extension supports the delayed rendering of data, which enables efficient synchronization of clipboards [1]. The interaction between the local clipboard and applications for Copy and Paste operations in the RDP connection is illustrated in Figure 1.

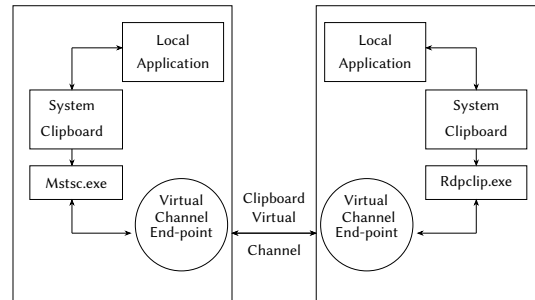


Figure 1: Copy and Paste operations through the clipboard virtual channel

During the “Copy” operation, when a local application on the client system copies data to the system clipboard, the virtual channel endpoint on the server receives a notification and updates the server’s clipboard with the same formats. Once the update is successful, the server acknowledges it.

During the “Paste” operation, the virtual channel endpoint on the server can send a clipboard data lock request to the client machine to prevent any changes to the data on the client machine’s system clipboard until an unlock request is sent. The local application on the server requests data from its own clipboard, and the server’s clipboard requests delay-rendered data from the virtual channel endpoint on the client machine. The virtual channel endpoint on the server sends a format data request for the requested data type, and the virtual channel endpoint on the client machine retrieves the data from the client’s system clipboard. The data is then returned to the virtual channel endpoint on the client machine, which sends it to the virtual channel endpoint on the server machine using a format data response. The virtual channel endpoint on the server machine updates the server’s clipboard with the received data, and the local application on the server receives the data from the server’s clipboard. Finally, the virtual channel endpoint on the server machine sends an optional unlock clipboard data protocol data unit (PDU).

Microsoft Terminal Services Client (Mstsc) is a Windows desktop application used by a client machine to create a remote desktop session on another machine [14]. On RDP Server, rdpclip.exe is responsible for all the related clipboard operations between the RDP server clipboard

and the RDP service. It is a normal process to interact with the RDP service via a dedicated virtual channel [6].

Remote access applications like TeamViewer, which provide users with the ability to remotely access and control a computer, also offer clipboard-sharing functionality. This feature allows for the seamless sharing of clipboard data between connected systems, making it easy to transfer information between remote and local computers. The process of sharing clipboard data via TeamViewer is similar to that used in RDP connections, where clipboard data is exchanged via a dedicated virtual channel.

In virtualized environments, such as VMware, two components facilitate the exchange of clipboard data: `vmtoolsd.exe`, which is responsible for interactions with the clipboard on the guest operating system, and `vmware-vmx.exe`, which is specific to the targeted Virtual Machine (VM) and resides on the host operating system. These components work together to coordinate the exchange of clipboard data between the guest and host operating systems. The process is illustrated in Figure 2.

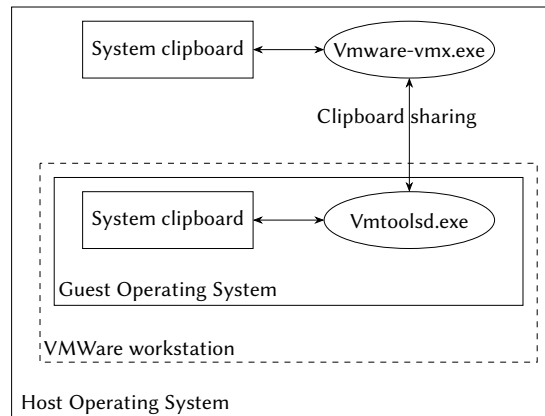


Figure 2: System clipboard interaction through virtualization.

In virtualized environments, transmitting changes to the clipboard to the other connection peer works similarly to the RDP case. Changes to the clipboard are transmitted in real-time, and clipboard data sharing is handled similarly to the way it is handled between `rdpclip.exe` and `mstsc.exe` in Windows RDP. However, changes made to the clipboard on either the guest or host are not automatically transmitted to the other side until the user moves the control to the other side to activate it.

4. Clipboard Data Attacks

4.1. Local Attacks

Clipboard hijacking and data manipulation attacks are sophisticated techniques that require in-depth analysis of the related Windows Application Programming Interface (API) calls. To simulate a clipboard data manipulator malware, a simple application can be created and run on a local Windows machine to analyze the interaction between the system clipboard and the machine. The goal is to demonstrate how clipboard data can be manipulated using standard Clipboard Windows APIs. By running this malware on a victim's machine, an attacker can use basic Windows API calls to manipulate the clipboard data. The process involves allocating memory space to hold the manipulation string, calling the `OpenClipboard` function to open the clipboard for examination and prevent other applications from modifying the clipboard content, and using the `EmptyClipboard` function to clear the clipboard and free handle to data in the clipboard. The `SetClipboardData` function is then used to place data on the clipboard in a specified clipboard format, which in this case is `CF_TEXT`. Finally, the `CloseClipboard` function is called to close the clipboard. Previous malware samples have utilized similar techniques for accessing clipboard data.

In their analysis, researchers in [15] identified multiple techniques used by malware authors to access clipboard data, including registering the malware as a clipboard viewer or hooking into the `SetClipboardData` and `GetClipboardData` functions to intercept and steal data from the clipboard. This can result in the theft of sensitive information, such as login credentials, credit card numbers, or personally identifiable information (PII).

To mitigate the risks associated with these attacks, it is important to implement appropriate security measures, such as clipboard data encryption [15], monitoring clipboard-related activities, and maintaining up-to-date antivirus software. By being vigilant and taking proactive steps to safeguard against these attacks, individuals and organizations can minimize the probability of potential compromise for their sensitive data.

4.2. Remote Attacks

In section 3, we discussed how different applications interact with the system clipboard and the RDP virtual channel extension. By monitoring the Windows API calls of `rdpclip.exe`, we identified the sequence of API calls that are invoked during clipboard data exchange between RDP peers. The primary functionality of clipboard data sharing is achieved through the use of various essential API calls, such as `AddClipboardFormatListener`, which registers a given window handler in the system-maintained clipboard format listener list. This

enables the handler to receive notifications when there is a change to the clipboard. Additionally, several functions from the OLE32.dll library, such as OpenClipboard, EmptyClipboard, and Multiple SetClipboardData, are responsible for opening the clipboard, emptying its contents, and setting data on the clipboard in different formats. Notably, the use of Delayed Rendering, whereby SetClipboardData with a NULL handler is called, allows for the manipulation of clipboard data by setting a given format but delaying the actual rendering of the data until requested by another application.

Based on our analysis, an attacker on a malicious remote machine, such as an RDP server, or malicious VM accessed through a VMWare console or a peer in a TeamViewer connection, can detect changes in clipboard data and manipulate or wipe the copied data with the knowledge of the data type on the client machine. The attacker can launch multiple attacks on the client machine once a connection is established with the malicious server or remote machine, as described in the previously stated cases. To avoid detection during memory forensics by investigators for the victim machine, the attacker could hook the SetClipboardData function instead of registering the malware as a clipboard viewer, as described in [16]. Extracting clipboard data from memory depends on traversing the Windows Station object, which has multiple attributes, including spwndClipViewer, which indicates if there is a registered clipboard viewer. Using the SetClipboardData function is a more stealthy approach than registering the malware as a clipboard viewer.

5. ATTACK IMPLEMENTATIONS

Windows clipboard hijacking, data manipulation, and sniffing attacks can be performed using the following approaches:

5.1. Host-based clipboard hijacking via malicious software

This attack involves the installation of malware on the victim's computer by the attacker. The malware is designed to monitor copying operations and compare clipboard data with predefined signatures for replacement. When the malware identifies a match, it proceeds to replace the identified data. To perform clipboard hijacking and data manipulation locally, the attacker must install malicious software on the victim's machine. This attack may require other tactics, such as social engineering, to deliver the malware [3].

Adding new patterns for manipulating targeted data may require an interactive connection with the attacker,

which increases the chances of detection and can be considered a limitation of the attack's success.

5.2. Remote clipboard hijacking via malicious RDP server

Once the victim connects to the malicious Windows RDP server, their clipboard is shared with the malicious server, which allows the attacker to monitor and manipulate any copy operations performed on the victim's machine. To carry out this attack, the attacker injects a malicious Dynamic Link Library (DLL) into the rdclip.exe process running on the RDP server. This DLL is used to hook the SetClipboardData API function, which enables the attacker to manipulate the clipboard data in real-time.

DLL injection is a technique that involves injecting code into a target application's memory space through a DLL, which allows for further interaction with the application's functions and memory [16]. Function hooking, on the other hand, is a method used to modify an application's behavior by forcing it to use a different function than it was initially intended to use [17].

It is worth noting that all proposed attacks and detection techniques in this study perform DLL injection in userspace.

To execute the attack, the attacker sets up a malicious Windows RDP server, which intercepts any data copied from the victim's machine when connected to the server. Function hooking is then used to manipulate the copied data on the clipboard, and the modified data is presented to the victim when he paste it. However, this attack blindly performs remote clipboard data manipulation, which can result in the replacement of any shared clipboard data with a fixed value or complete deletion, leading to a denial of service for copy-and-paste operations on the victim's machine.

The attack flow chart is illustrated in Figure 3.

5.2.1. Attack walkthrough

The primary goal of this attack demonstration is to showcase the impact of modifying clipboard data during a remote desktop session. To achieve this objective, the attack employs the technique of DLL injection, which is not confined to RDP sessions only but also extends to other remote access applications such as TeamViewer and VMWare console.

The initial step of this attack process involves the injection of a malicious DLL into the intended process. Depending on the application in question, the target process may differ; for instance, in the case of a Windows RDP connection, the target process is rdclip.exe, whereas, for TeamViewer, it is teamviewer.exe. Similarly, for VMWare, the target process is vmtoolsd.exe which provides all virtual machine guest tools. The process injection can occur

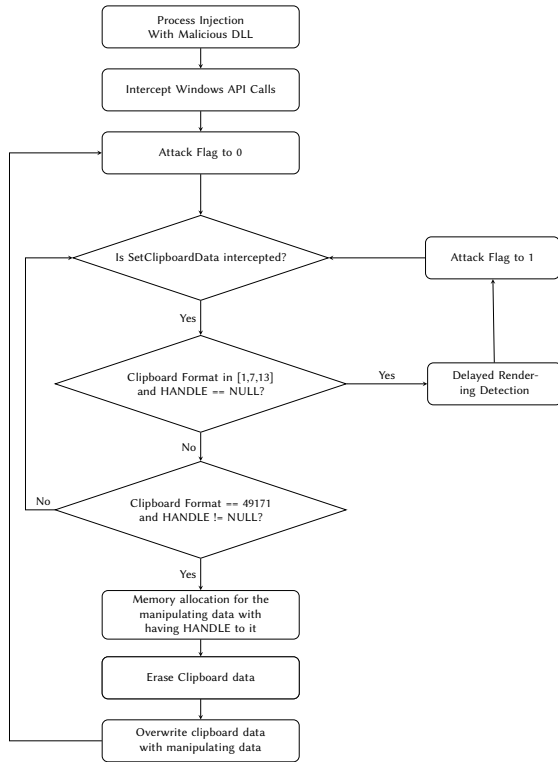


Figure 3: Remote clipboard data manipulation attack Flow chart.

either before or after the connection is established, and various techniques can be utilized for DLL injection. For this purpose, a simple program has been developed, although multiple methods can be employed, including Microsoft Detours [18].

After injecting the DLL into the targeted process, API calls to the `SetClipboardData`, when a copy operation is performed, `SetClipboardData` function is intercepted, and one of two conditions is met: retrieving the Clipboard Format with NULL Handler (Delayed Rendering), or retrieving the Clipboard Format with a handler to the OLE Private Data (Immediate Rendering). In the latter case, a memory area is reserved for the manipulation string, and the clipboard data is erased by calling `EmptyClipboard`. Then, `SetClipboardData` is called with `CF_TEXT` as the Clipboard Format and a handle to the reserved memory holding the manipulation string. The function is called again, and this time, it returns a handle to hold a pointer to the real function, which is executed successfully.

While the above process manipulates the clipboard data successfully, it does not differentiate between copying text or copying a file. Consequently, in the case of copying a file, the clipboard data is manipulated, and the

held data is the manipulation text. To address this issue, a flag to detect both delayed rendering and setting data cases is required.

This attack can remotely manipulate any copied text on the victim's machine with pre-hardcoded text, as a proof of concept. However, it can be modified to target a specific pattern of copied text, such as a cryptocurrency wallet ID. This could be achieved by retrieving the original copied text, comparing it with the targeted pattern, and then performing the manipulation if there is a match.

5.3. Remote Clipboard Data Targeted Manipulation

The objective of this attack is to manipulate specific shared clipboard data patterns during a remote desktop session, such as replacing a website URL with a phishing website or replacing a cryptocurrency wallet ID with the attacker's ID. While similar to the previous attack, this one uses a different technique. Instead of hooking the `SetClipboardData` function, it detects a specific sequence of Windows-related API clipboard calls. To achieve this, all the included API calls are hooked, and a counter is incremented once a targeted API call is detected in the correct sequence (see Algorithm 1). The counter is reset to zero once the last API call in the targeted sequence is detected.

Algorithm 1: Targeted API Sequence Detection

Function `HookedOpenClipboard()`:

```

if APICounter == 0 then
  | APICounter = APICounter + 1;
  return RealOpenClipboard();

```

The targeted sequence consists of `OpenClipboard`, `EmptyClipboard`, `SetClipboardData`, and `CloseClipboard`. `OpenClipboard` is called to lock other applications from modifying the clipboard contents, followed by `EmptyClipboard` to erase the clipboard data contents, and then `SetClipboardData` to set the actual data into the clipboard after the Delayed Rendering occurs, as discussed earlier. `CloseClipboard` is called to commit the clipboard data changes and release the lock that was created when `OpenClipboard` was called. Finally, `GetClipboardOwner` is called to retrieve the current owner of the clipboard by returning a handle to it. Remote clipboard data manipulation is performed once this sequence of API calls is detected during the hooking of `GetClipboardOwner`. This is done by retrieving the clipboard data using `OpenClipboard` followed by `GetClipboardData` with `CF_TEXT` as an argument to retrieve the clipboard data in text format. The targeted data pattern is compared to the retrieved data, and if

Algorithm 2: Remote Clipboard Data Manipulation

```
Function HookedGetClipboardOwner():  
  if APICounter == 5 then  
    RealOpenClipboard(NULL); dataHandle ← GetClipboardData(CF_TEXT);  
    data ← GetDataFromHandle(dataHandle); matched ← FindPatternMatch(data);  
    if matched ≠ FALSE then  
      RealEmptyClipboard();  
      manipulationHandle ← GlobalAlloc(GMEM_MOVEABLE, len(manipulationText));  
      RealSetClipboardData(CF_TEXT, manipulationHandle);  
      RealCloseClipboard();  
      APICounter = 0;  
  return RealGetClipboardOwner();
```

there is a match, `EmptyClipboard` is called to erase the clipboard data, followed by `SetClipboardData` to set the clipboard data with the malicious content. `CloseClipboard` is then called to release the lock after the manipulation has been performed, as shown in Algorithm 2.

To prevent recursion between the targeted API calls used for detecting the targeted API calls and the attack itself, the real function points are called instead of the hooked function points, similar to the previous attack.

We have applied this attack successfully to Microsoft RDP, remote access applications such as TeamViewer, virtualization software such as VMWare, and other applications that share the clipboard with the same technique used by Microsoft RDP.

5.4. Remote Clipboard Data Sniffing

In this variation of the attack, a malicious malware is designed to intercept and log all text copied on a victim’s machine through a remote connection, such as RDP or any other vulnerable application that enables clipboard sharing. Similar to the previous attack, this one also targets a specific sequence of Windows API calls related to the clipboard. However, upon detecting the sequence of targeted API calls, the data is simply copied and logged on the malicious remote server, unlike the previous attack that aimed to manipulate shared clipboard data. The targeted API calls start with `OpenClipboard`, followed by `EmptyClipboard`, `SetClipboardData` twice for `Delayed Rendering` and `actual data setting`, followed by `CloseClipboard` to release the clipboard lock, and finally `GetClipboardOwner` is called. Logging the captured shared clipboard data is performed by calling a traditional sequence of API calls starting with `OpenClipboard`, followed by `GetClipboardData` to retrieve the victim’s shared clipboard data, and finally releasing the clipboard lock by calling `CloseClipboard`, as shown in Algorithm 3.

Additionally, this attack can affect victim machines

with any version of Microsoft Windows operating systems, as long as they have an active connection with the malicious server through RDP, Remote Access applications like TeamViewer, virtual machines hosted by VMWare and accessed through VMWare console, or any other application that shares clipboard data using the same technique as RDP.

6. ATTACK DETECTION

In the context of ensuring computer system security, distinguishing between benign and malicious clipboard operations in remote connections is a critical task. This requires identifying a specific sequence of Windows API clipboard calls that occur during the copying of text on a remote Windows server accessed through RDP or a VMWare console or a remote access application like TeamViewer.

To detect such sniffing, the first step is to identify the format of the clipboard data using the `GetClipboardFormatNameW` function, followed by detecting the `OpenClipboard`. The next step is to monitor the `GetClipboardData` and finally the `CloseClipboard` function to indicate the attacker’s completion of accessing the clipboard data. To detect the sniffing over TeamViewer, it requires detecting the previous sequence in addition to detection `OpenClipboard` followed by `GetClipboardData` and finally `CloseClipboard`.

For blind remote clipboard data manipulation, the required sequence of API calls is shown in Figure 4.

To detect targeted manipulation of remote clipboard data via RDP, a sequence of Windows API calls have to be monitored. The process involves retrieving the format name of the clipboard data, followed by retrieving the clipboard data to compare it with targeted data patterns. Then the clipboard is closed and two formats, `CanIncludeInClipboardHistory` and `CanUploadToCloudClipboard`, are registered [19]. After reopening the clipboard and emptying it, the clipboard data is set twice and the clipboard is reclosed. Fi-

Algorithm 3: Remote Clipboard Data Sniffing

```
Function HookedGetClipboardOwner():  
  if APICounter == 5 then  
    RealOpenClipboard(NULL); dataHandle ← GetClipboardData(CF_TEXT);  
    data ← GetDataFromHandle(dataHandle); LogSniffedData(data);  
    RealCloseClipboard();  
    APICounter = 0;  
  return RealGetClipboardOwner();
```

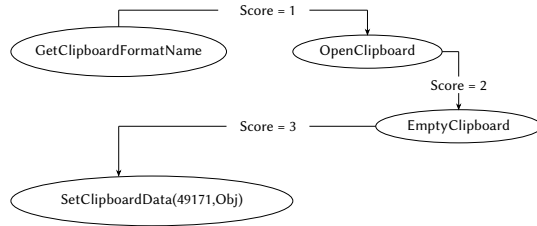


Figure 4: Malicious remote Windows API calls.

nally, the previous sequence of opening and closing the clipboard is repeated to flush the clipboard data and set the manipulated data. Attackers utilize this sequence of functions to manipulate the clipboard data, making it essential to detect these functions in sequence to prevent data hijack and manipulation.

A flagging mechanism can be employed to effectively detect the sequence of Windows API calls for remote clipboard data sniffing or targeted manipulation. The mechanism involves setting and incrementing a flag each time a targeted API is detected in the correct order. Once the last targeted API call is detected and the flag threshold is reached, an alert is triggered to notify the user or system administrator. To minimize false positives, the time factor, i.e., the time difference between the first and last detected API calls in the targeted sequence, should be measured.

Integrating the timing factor with other detection mechanisms can differentiate between normal remote clipboard data copying and targeted attacks. The study presents the necessary time for detecting the targeted sequence of API calls for each attack, serving as a valuable reference for devising effective detection and prevention strategies. However, relying solely on detecting the proposed targeted sequences of API calls to detect remote clipboard data sniffing or manipulation may lead to false positives. Therefore, the time factor can aid in differentiating between the two scenarios. Table 2 provides the results of calculating the required time for the attack to be performed.

In the case of an attack initiated from a virtual machine accessed via VMWare console, the timing factor may not

be effective in detecting remote clipboard data sniffing or targeted manipulation. This is because the exchange of clipboard data in this scenario depends on switching between the host and guest operating systems. Therefore, the timing for detecting the targeted sequences of API calls may not indicate an attack in progress.

7. RESULTS EVALUATION

The section presents the results of the clipboard data attacks proposed in this study and evaluates the effectiveness of the suggested detection techniques. This evaluation process is crucial for assessing comprehensively the security of computer systems against such attacks and for devising more robust security protocols. To ensure a comprehensive evaluation, all test cases considered in this study assume that the malicious server is based on Microsoft Windows since the Remote Desktop Protocol is a Microsoft protocol. The tests included clients with various operating systems, and including Windows 7, Windows 10, and Windows 11 with both x86 and x64 versions. The results presented in this section provide critical insights into the vulnerabilities of sharing clipboard data and the effectiveness of the proposed detection techniques in detecting and preventing potential security threats. Through the analysis and discussion of the results, this study aims to advance our understanding of clipboard data security and guide the development of more effective security measures.

7.1. Attack Results

This section outlines the results obtained from conducting remote clipboard data attacks using various test case scenarios as listed in Table 3. The experiments were carried out using different versions of Microsoft Windows including Windows 7, Windows 10, and Windows 11 with both x86 and x64 versions. These findings suggest that any remote access application that shares clipboard data using the discussed technique would be susceptible to the proposed attacks.

The present study conducted experiments to evaluate the effectiveness of the proposed remote clipboard data attacks, as listed in Table 3. The results of all test

Table 2

Remote clipboard data manipulation attacks required time.

Remote access application	Attack	Spent time (Seconds)
Native Windows RDP	Sniffing and Manipulation	$0 < T < 0.1$
TeamViewer	Blind Manipulation	$0.1 < T < 0.5$
TeamViewer	Sniffing	$0.1 < T < 0.7$

Table 3

Remote clipboard data manipulation attacks and detection results.

Remote Connection Application	Attack Results	Detection Results
Windows RDP	Succeeded	Detected
VMWare Console	Succeeded	Not Detected
TeamViewer	Succeeded	Detected

cases showed successful attacks. When a victim connects to a malicious server that has a malicious DLL for the proposed blind manipulation attack, his clipboard data is compromised. The copied data is either modified or wiped out for as long as the victim's connection with the malicious server is active.

For the remote clipboard data sniffing attack, the victim's copied data is immediately logged on the remote server after a copy operation is performed on the victim's machine. Lastly, the attack that aimed to manipulate specific shared clipboard data monitored all shared copied data on the victim's machine and performed the manipulation when a match occurred, without affecting any other copied data. These findings illustrate the vulnerabilities of sharing clipboard data and emphasize the importance of implementing robust security measures to prevent such attacks.

7.2. Detection Results

This section presents the results of the evaluation of the proposed detection techniques for clipboard data attacks. The evaluation was conducted using a range of test cases, assuming a malicious server based on Microsoft Windows, as the Remote Desktop Protocol is a Microsoft protocol. Tests included clients with various versions of Windows including Windows 7, Windows 10, and Windows 11 with both x86 and x64 versions. The results provide valuable insights into the effectiveness of the proposed detection techniques in detecting and preventing potential security threats related to clipboard data sharing.

The presented detection method was effective in detecting all the cases mentioned in Table 3, except when the victim accessed a virtual machine armed with the attacking tool through the VMWare console, as it is a limitation in our proposed detection technique.

8. FUTURE WORK

One promising avenue for future research is the examination of clipboard attacks that focus on copied files. Such attacks could involve the insertion of malicious code into an executable file that a user copies, which could subsequently be executed on the victim's device. The techniques outlined in this paper could be used to execute this type of attack, with modifications made to detect the copying of executable files and insert the malicious code into the targeted file. Further exploration of this area could yield valuable insights into the potential dangers posed by these attacks and facilitate the development of stronger defense mechanisms.

9. CONCLUSION

This research paper provides a comprehensive examination of the security risks associated with clipboard data sharing across various environments. The investigation considers clipboard data sharing on local machines, Remote Desktop Protocol (RDP) sessions, and virtualized environments and identifies remote clipboard data manipulation as a significant threat to security. The paper presents detailed explanations of the different types of

attacks that can occur on shared clipboard data, with a focus on remote clipboard data manipulation and sniffing. The efficacy of proposed detection and prevention techniques is evaluated through successful experiments that demonstrate the importance of being vigilant against potential clipboard data attacks. This research contributes to the understanding of clipboard data security and emphasizes the need for further research and development of more robust security measures to protect against attacks.

References

- [1] Microsoft, [MS-RDPECLIP]: Remote Desktop Protocol: Clipboard Virtual Channel Extension, [Online]. Available: <https://winprotocoldoc.blob.core.windows.net/productionwindowsarchives/MS-RDPECLIP/%5BMS-RDPECLIP%5D.pdf>, 2021.
- [2] M. Manna, A. Case, A. Ali-Gombe, G. G. Richard III, Memory analysis of .net and .net core applications, *Forensic Science International: Digital Investigation* 42 (2022) 301404.
- [3] A. Pillai, V. Saraswat, A. VR, Smart wallets on blockchain—attacks and their costs, in: *Smart City and Informatization: 7th International Conference, iSCI 2019, Guangzhou, China, November 12–15, 2019, Proceedings 7*, Springer, 2019, pp. 649–660.
- [4] G. S. GBHackersX: Metamask - first copy-and-paste hijacking crypto malware found in google play, <https://gbhackers.com/clipper-hijacking-malware/>, 2019.
- [5] L. Abrams, Clipboard Hijacker Malware Monitors 2.3 Million Bitcoin Addresses, <https://www.bleepingcomputer.com/news/security/clipboard-hijacker-malware-monitors-23-million-bitcoin-addresses/>, 2018.
- [6] E. Itkin, D. Baril, He said, she said - poisoned rdp offense and defense, 2019. URL: <https://i.blackhat.com/USA-19/Wednesday/us-19-Baril-He-Said-She-Said-Poisoned-RDP-Offense-And-Defense-wp.pdf>, black Hat.
- [7] Z. Wang, X. Wu, C. Liu, Q. Liu, J. Zhang, Ransomtracer: exploiting cyber deception for ransomware tracing, in: *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, IEEE, 2018, pp. 227–234.
- [8] J. Woodruff, J. Alexander, Data transfer: A longitudinal analysis of clipboard and drag-and-drop use in desktop applications, *International Journal of Human-Computer Studies* 132 (2019) 112–120.
- [9] J. Okolica, G. L. Peterson, Extracting the windows clipboard from physical memory, *digital investigation* 8 (2011) S118–S124.
- [10] Microsoft, About the clipboard, Microsoft, 2021. URL: <https://docs.microsoft.com/en-us/windows/win32/dataxchg/about-the-clipboard>.
- [11] Microsoft, Clipboard, Microsoft, 2021. URL: <https://docs.microsoft.com/en-us/cpp/mfc/clipboard?view=msvc-170>.
- [12] Microsoft, Clipboard: When to use each clipboard mechanism, Microsoft, 2021. URL: <https://docs.microsoft.com/en-us/cpp/mfc/clipboard-when-to-use-each-clipboard-mechanism?view=msvc-17>.
- [13] Microsoft, Clipboard operations, Microsoft, 2022. URL: <https://docs.microsoft.com/en-us/windows/win32/dataxchg/clipboard-operations#delayed-rendering>.
- [14] Microsoft, mstsc, Microsoft, 2021. URL: <https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/mstsc>.
- [15] G. Pathak, G. K. Tak, Implementation of clipboard security using cryptographic techniques, *International Journal of Computer Applications* 86 (2014).
- [16] J. Berdajs, Z. Bosnić, Extending applications using an advanced approach to dll injection and api hooking, *Software: Practice and Experience* 40 (2010) 567–584.
- [17] J. Lopez, L. Babun, H. Aksu, A. S. Uluagac, A survey on function and system call hooking approaches, *Journal of Hardware and Systems Security* 1 (2017) 114–136.
- [18] G. Hunt, D. Brubacher, Detours: Binary interception of win 32 functions, in: *3rd usenix windows nt symposium*, 1999.
- [19] Microsoft, Clipboard formats, Microsoft, 2020. URL: <https://learn.microsoft.com/en-us/windows/win32/dataxchg/clipboard-formats>.