Malware Analysis on "tnnbtib.exe"

Bipul Paudel
Dept. of Computing and Engineering
of George Mason University
Virginia, United States
bpaudel2@gmu.edu

***Abstract*— This research paper comprehensively analyzes the malware 'tnnbtib.exe,' utilizing static analysis, dynamic analysis, and advanced reverse engineering techniques. The study focuses on understanding the malware's behavior, functionalities, and impact on infected systems.**

Keywords—static, dynamic, reverse engineering, IDA pro, system internal suite, PEID, Olly debugger, sha-256 hash, TCP View, strings, virtual machines, UPX, packed, Wireshark, hex, entry point, entry point, virtual address, MITRE ATT&CK Framework, DLL, .data, .idata, .bss, .text, packers, CPU, registry, DNS, HTTP

I. INTRODUCTION

In an era where digital threats are increasingly sophisticated, this paper aims to dissect the 'tnnbtib.exe' malware using a blend of static, dynamic, and reverse engineering methods, offering insights into its operational tactics and potential threats. Our analysis delves into the intricate architecture and behavior of this malware, chosen for its complexity and relevance in today's cybersecurity landscape. By meticulously examining the malware's code structure and behavior both in a non-executable state and during execution, we seek to uncover its capabilities, strategies for evasion, and impact on infected systems. Adhering to strict technical frameworks and differentiating between factual findings and speculative conclusions, this research contributes a comprehensive understanding of 'tnnbtib.exe', aiming to enhance the knowledge of this particular malware and the affected system.

II. METHODOLOGY

In our comprehensive analysis of the 'tnnbtib.exe' malware, we employ a structured methodology that segregates the process into distinct phases of static analysis, dynamic analysis, and reverse engineering, utilizing a specific set of tools in each phase to unravel the malware's complexities. All analytical procedures are meticulously conducted within a secure, isolated virtual environment. This setup utilizes Windows XP as the operating system, effectively virtualized using VMware's platform to ensure a controlled and safe analysis of the malware, free from external interferences.

A. Static Analysis

The static analysis phase forms the foundation of our investigation, where we initially examine the malware without executing it. For hex analysis, HxD is our primary tool, allowing us to delve into the malware's binary data at a granular level. This hex editor facilitates an in-depth view of the raw hex, enabling us to uncover hidden structures and potentially obfuscated segments within the malware.

For basic file analysis, we use the File tool to gather essential information about the malware, including file type, size, and hash values. This information is crucial for the initial classification and integrity verification of the malware sample. strings.

PEID is employed to extract and analyze relevant strings from the malware. This tool is instrumental in identifying the packers and compilers used, providing early insights into the malware's obfuscation and packing methods. Additionally, PEID aids in the preliminary identification of the malware's characteristics and behavior patterns based on the extracted strings.

Virustotal is utilized to cross-reference the malware against a vast database of known malware signatures. This online service provides a comprehensive scan using multiple antivirus engines, offering insights into how the malware is detected and classified by different security solutions.

B. Dynamic Analysis

In the dynamic analysis phase, tools from the System internal suite, such as Process Explorer and Process Monitor. These tools are used to monitor the malware's behavior in a controlled execution environment. Process Explorer provides real-time monitoring of system processes and resources, while Process Monitor logs file system and registry offering a detailed view of the malware's actions.

TCP View is employed to observe the malware's network interactions, tracking its communication patterns and connections. Wireshark, a network protocol analyzer, is also used for capturing and analyzing network traffic, providing an in-depth look at the malware's network communication strategies and data exfiltration methods.

C. Reverse Engineering

In the reverse engineering phase, we delve deeper into the malware's internal workings. IDA Pro is a cornerstone in this stage, serving as an advanced disassembler and debugger. It allows us to break down the malware's executable code into a readable format, revealing the execution flow and interactive elements within the system. This tool is essential for a detailed understanding of the malware's architecture and the logic embedded within its code.

Olly Debugger complements IDA Pro in the reverse engineering process. It provides a dynamic analysis environment where we can observe the malware's execution in real time. This debugger is particularly useful for examining the runtime behavior of the malware, tracking its system interactions, and identifying key operational moments and changes.

In cases where the malware employs UPX packing, detected by tools like PEID, UPX unpackers are used to reverse the packing process, revealing the malware's original state. This de-obfuscation step is vital for a thorough understanding of the malware's code and functionality, as we will learn later in this research.

III. Static Analysis on "tnnbtib.exe"

Static analysis serves as a foundational approach in malware reverse engineering, offering a non-intrusive means to dissect and understand malware. This method is essential for revealing the malware's inherent architecture, functions, and potential impact in a safe, controlled environment, without the risk of execution.

Our analysis commenced with an examination of the malware file's basic properties. The file tool aided in the discovery of a portable executable with a size of 8329 bytes. Its SHA-256 hash value (E1A4B00D24AEB67FEC1341103F94E1256D4CB5A696DD02847D9EADB60BE4EC55) serves as a unique identifier, crucial for verifying the file's integrity and singularity in subsequent stages of the analysis.

A. String Analysis

In the string analysis phase, various strings such as KERNEL32.DLL, CRTDLL.DLL, LoadLibraryA, GetProcAddress, WININET.DLL, ADVAPI32.DLL, USER32.DLL, and wsock32.dll were extracted. These strings are indicative of the malware's interaction with essential system libraries[2], hinting at its capabilities in manipulating system processes, interfacing with network protocols, and possibly engaging in user activity monitoring. This analysis of strings is a key step in hypothesizing the malware's functionality and potential attack vectors which we will investigate further in the reverse engineering portion of the research.

B. Signature Analysis

Signature analysis using Virustotal [3] provided further insights, identifying the malware as part of the Slack Bot, Didler, and Slack malware families. Interestingly, it also identified that 61 security vendors flagged this malware as malicious [3] however, no sandboxes flagged the file as malicious, suggesting a level of sophistication in evasion techniques.

C. Binary Analysis

The binary analysis phase involved disassembling the malware to understand its structure and code. This stage is pivotal in identifying significant functions and algorithms, such as those used for encryption, network communication, and data exfiltration. The disassembly process unraveled the complexities of the malware's code, providing a clearer view of its operational mechanics which we will revisit when we reverse engineer the program.

D. Obfuscation techniques

Obfuscation techniques were evident in the malware's use of UPX packers, as identified by specific sections named UPXA, UPX1, and UPX2 during the hex analysis. The deobfuscation process is done utilizing the UPX tool which we can easily use to reveal the underlying unpacked code and structure which we will also later explore in grave detail.

Dependency analysis revealed the malware's reliance on various DLLs we extracted from string analysis, each playing a different role in its operation. These dependencies are critical for the malware's interaction with the host system, encompassing functionalities related to system processes, network communications, and user interface manipulation.

E. Hex Analysis

Detailed hex analysis was crucial in deepening our understanding of the malware's technical composition. By examining specific aspects such as the CPU type, Image base, Entry point, and Section addresses, we gained a granular view of its structure. This approach not only contextualized the malware within the broader landscape of known threats but also highlighted its unique attributes. The outputs of the hex analysis are as follows:

- CPU Type: 014C, indicating the specific processor architecture used by the malware.
- Image Base: 0x400000, showing the default location in memory where the executable is loaded.
- Entry Point: 8760h (0x408760), marking the starting point of the program in the memory.

TABLE I. SECTIONS AND VIRTUAL ADDRESSES

Section Name	Virtual Address
UPXA	0x00001000
UPX1	0x00007000
UPX2	0x00009000

These specific details from the hex analysis not only provide insight into the malware's operational framework but also aid in identifying potential vulnerabilities and defense strategies against it.

F. *Conclusions from Static Analysis*

In conclusion, the static analysis of 'tnnbtib.exe' systematically deconstructed the malware, uncovering its structural and operational characteristics. These insights are instrumental for comprehending its capabilities and potential impact. The tools employed, notably HxD, PEID, and File, were critical in this exploration. As we transition to the dynamic analysis phase, the knowledge gained here will guide our analysis, helping us understand the malware's behavior during execution and paint a complete picture of its functionality and threat potential.

IV. DYNAMIC ANALYSIS ON "TNNBTIB.EXE"

Dynamic analysis stands as a pivotal component in this research, offering insights into the real-time behavior of malware as it interacts with the system. This method is vital for observing the malware's operational characteristics during execution, including its interactions with system processes, network activities, and modifications to the host environment. Unlike static analysis, which scrutinizes the malware's dormant code, dynamic analysis sheds light on its active behavior, providing a more comprehensive understanding of its capabilities and intentions.

A. *Setup of the Dynamic Analysis Environment*

For this analysis, we meticulously created a controlled, isolated environment to safely observe the malware's actions. A Windows XP system was selected for its relevance to the malware's targeted vulnerabilities and was virtualized using VMware. This setup, isolated from the host operating system and running on a NAT network, provided a secure, controlled space for our investigation, allowing us to monitor the malware's behavior without risking broader network exposure or system compromise.

B. *Execution and Behavioral Monitoring*

Our dynamic analysis commenced with the execution of 'tnnbtib.exe'. Utilizing tools like PEID for preliminary analysis, we identified the packing methods employed by the malware. Process Monitor and Process Explorer were critical in providing a real-time view of system interactions, tracking every file system activity and registry change initiated by the malware. TCPView played a key role in monitoring network connections, offering a window into the malware's communication with external servers. Additionally, Wireshark was utilized in order to capture and analyze network traffic, particularly or detect any data exfiltration attempts or command and control communications.

C. *Observations*

The dynamic analysis revealed significant alterations to the Windows registry, a common tactic for achieving persistence and modifying system behavior. Key registry changes included:

- Registry Modifications HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings, indicating attempts to alter internet configurations and potentially disable security settings.
- Registry Changes in HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run, suggesting efforts to ensure the malware's automatic execution upon system startup, a tactic commonly used for maintaining persistence.
- Registry Alterations to HKCU\Software\Microsoft\windows\CurrentVersion\Internet Settings\Connections, likely aimed at manipulating network connections and settings.

These registry modifications are crucial indicators of the malware's intent to embed itself within the system and manipulate its behavior to suit its objectives.

D. *Evasion Techniques*

A key observation in the dynamic analysis was the malware's creation of a system's registry under, "HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Run" which was discovered using process monitor. This particular registry path is critical as it enables programs to auto-

start upon system bootup. The malware's use of this registry key demonstrates a sophisticated evasion technique, allowing it to surreptitiously establish persistence on the host machine. By embedding itself in this auto-start location, 'tnnbtib.exe' ensures it remains active and potentially undetected by the user, executing each time the system is restarted.

The significant registry modifications mentioned earlier include alterations in internet settings and startup processes, indicating efforts to modify system configurations and establish persistence.

E. Analyzing Network Behavior and Impact

A critical aspect of our dynamic analysis of 'tnnbtib.exe' involved scrutinizing its network behavior, which revealed significant insights into the malware's operational intent and potential impact. Using network monitoring tools like TCPView and Wireshark, we observed the malware establishing connections to several external IP addresses and domains, indicative of its communication strategy (and potential remote-control capabilities.)

Among the notable IP addresses, the malware connected are as follows:

TABLE II. DNS NETWORK ACTIVITY

DNS Network Activity

IP	Port
234.151.42.104	53
83.188.255.52	53
137.90.64.13	53

These connections raise concerns about the purposes of these communications, which could range from receiving further instructions from a command-and-control server to exfiltrating sensitive data from the infected system. The malware also made HTTP requests which are as follows:

TABLE III. HTTP NETWORK ACTIVITY

Connections	Port
http://sb.webhop.org	80
http://irc.slim.org.au	80
malwarecourse.sans.org	80

(which could be involved in command and control or data exfiltration activities)

The diversity and nature of these network communications underscore the sophistication of 'tnnbtib.exe', highlighting its potential for engaging in activities such as data theft, espionage, and remote system manipulation. The ability to communicate with external servers and execute commands or transmit data underscores the significant threat posed by this malware.

V. MITRE ATT&CK TECHNIQUES

In the dynamic analysis of 'tnnbtib.exe', we observed several behaviors aligned with the MITRE ATT&CK framework [6], a globally accessible knowledge base of adversary tactics and techniques. This framework provided a structured classification of the malware's activities, offering deeper insights into its strategies and potential threats.

The malware demonstrated significant capabilities in Privilege Escalation (TA0004) and Process Injection (T1055), evident from its ability to spawn processes, potentially allowing it to execute code with elevated privileges. This is a critical aspect of its operation, enabling the malware to manipulate system processes and potentially gain unauthorized access to system resources.

Defense Evasion techniques (TA0005) were also prominent. The malware employed Obfuscation (T1027), being packed with UPX, a common packing tool. This technique, known as Software Packing (T1027.002), complicates the analysis and detection of the malware, allowing it to evade security measures. Additionally, the malware engaged in Masquerading (T1036) by dropping PE files into the Windows directory and executing them. This tactic is designed to conceal its true nature and intentions, blending in with legitimate system files.

A notable aspect of the malware's behavior was its use of Virtualization/Sandbox Evasion (T1497). It exhibited patterns indicative of evasive loops, potentially designed to detect, and respond to analysis environments. Such techniques can significantly hinder dynamic analysis, showcasing the malware's advanced capabilities to avoid detection.

In the realm of Discovery (TA0007), the malware performed actions aligned with Remote System Discovery (T1018) and System Information Discovery (T1082). It read the host's file and software policies, activities that likely provided it with valuable information about the system and network environment, aiding in further malicious activities or lateral movement.

Lastly, under Command and Control (TA0011), the malware utilized both Application Layer Protocol (T1071) and Non-Application Layer Protocol (T1095) tactics, as evidenced by its DNS lookups. These activities are typically indicative of the malware communicating with external command and control servers, a critical component for receiving instructions or exfiltrating data.

In summary, the dynamic analysis of 'tnnbtib.exe' revealed a range of sophisticated techniques that align with the MITRE ATT&CK framework. Understanding these tactics and techniques is essential for developing effective countermeasures and understanding the full scope of the malware's capabilities and threats.

VI. Reverse Engineering Methods

Reverse engineering in malware analysis is an essential process that involves deconstructing a malware's compiled binary back into a more understandable form, such as source code or assembly language. This approach is pivotal for comprehending the intricate workings of a malware specimen that cannot be fully understood through static or dynamic analysis alone. It allows us to delve deep into the malware's architecture, revealing its true functionality, operational mechanisms, and potential impact. By disassembling the code, setting breakpoints, and stepping through the instructions, reverse engineering provides a granular view of the malware's behavior, offering insights into its objectives, tactics, and the sophistication of its design.

For 'tnnbtib.exe', reverse engineering is particularly significant due to its UPX-packed nature, a common obfuscation technique used by malware authors to evade detection and analysis. Unpacking 'tnnbtib.exe' using the UPX tool and the '-d' command is the first critical step in this process, as it reveals the underlying code that was previously concealed. This unpacking process is like peeling back layers of an onion as shown in Fig. 3 [4], allowing us to access the true essence of the malware. The subsequent in-depth investigation using tools like IDA Pro enables a thorough examination of the malware's operational flow and functionalities. Through this meticulous process, reverse engineering empowers us to uncover the hidden mechanisms of 'tnnbtib.exe', laying down its tactics and strategies, and forming the basis for developing effective countermeasures.

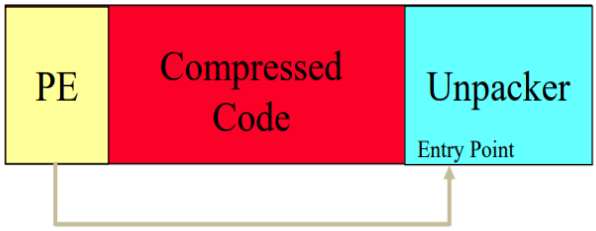

Fig. 1. *Visualization of a packed program on disk [4]*

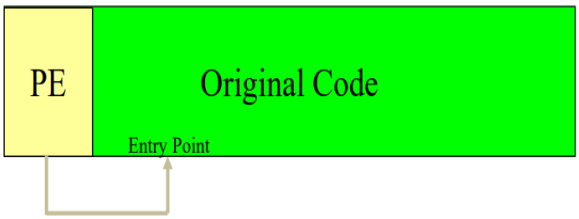

Fig. 2. *Visualization of an unpacked program on disk[4]*

After unpacking the malware and opening it in a hex editor (HxD), we can see significant changes in the characteristics of this portable executable, such as hash change(1A28360CF76D1FAE638C6F28D08EFB5FD24DB9557F6725AF08D3128B44EEFBAF). One major difference we see is the number of sections and the names of the mentioned sections. The sections now appear as .text, .bss, .data, and .idata. The contents of the individual sections are as follows:

TABLE IV. Executable Code Sections

Sections	Definition
.text	All code segments reside in this single section [4]
.bss	Represents uninitialized data for the application including all variables declared as static with a function [4]
.data	All application or module global variables are stored here [4]

.idata	The .idata section contains various information about imported functions including the Import Directory and Import Address Table.[4]

Along with the updated sections, we notice there is a new entry point for this unpacked binary. The new entry point resides in 11CBh (0x4011cb). The PE also consists of new virtual addresses for the sections which are as follows:

TABLE V. SECTIONS AND VIRTUAL ADDRESSES OF UNPACKED FILE

Section Name	Virtual Address
.text	0x00001000
.bss	0x00004000
.data	0x00005000
.idata	0x00006000

Unpacking the binary of 'tnnbtib.exe' using the UPX tool effectively strips away the layers of obfuscation, unveiling the malware in its raw assembly form. This transformation is crucial as it transitions the malware from a foggy, packed state into a transparent format, ready for detailed examination. In this newly revealed state, the assembly code becomes a rich source of information, laying the groundwork for a more in-depth and precise analysis. Utilizing IDA Pro, a powerful disassembler and debugger, we can meticulously navigate through this assembly code to locate the critical .idata section.

The .idata section, or Import Directory Table, is a special place for insights, revealing the specific Dynamic Link Libraries (DLLs) that the malware relies upon. By examining this section with IDA Pro, we gain a valuable understanding of the malware's external dependencies and interactions. The DLLs listed in this section are indicative of the functionalities that the malware seeks to exploit or manipulate within the host system. For instance, dependencies on certain systems or network-related DLLs can hint at the malware's capabilities in terms of network communication, data manipulation, or system monitoring. This level of analysis is instrumental in piecing together the malware's operational framework, providing a clearer picture of its intended behavior and potential impact. Analyzing the import section with IDA Pro thus becomes a critical step in comprehensively understanding 'tnnbtib.exe' and its underlying mechanics. The imports and their definitions are as follows:

TABLE VI. KEY IMPORTS FROM .IDATA AND DEFINITIONS

Sections	Definition
.text:00401497 push offset ModuleName ; "KERNEL32"	This DLL exports functions related to process, memory, hardware, and filesystem operations. Malware imports API functions from these DLLs to carry out filesystem-memory and process-related operations. [1]
.idata:004062EC ; Imports from ADVAPI32.DLL	This contains functionality related to service and registry. Malware uses the API functions from this DLL to carry out service-and-registry-related operations. [1]
.idata:00406314 ; Imports from CRTDLL.DLL	It usually contains a set of procedures and driver functions, which may be applied by Windows. [2]
.idata:004062E0 ; Imports from USER32.DLL	It implements functions that create and manipulate Windows user interface components, such as the desktop, windows, menus, message boxes, prompts, and so on. Some malware programs use functions from this DLL for performing DLL injections and for monitoring keyboard (for keylogging) and mouse events. [1]
.idata:00406254 ; Imports from WININET.DLL	It exposes high-level functions to interact with HTTP and FTP protocols. [1]
.idata:00406220 ; Imports from wsock32.dll	They contain functions for communicating on the network. Malware imports functions from these DLLs for performing network-related tasks. [1]

A. *Conclusion of Reverse Engineering Methods*

The reverse engineering of 'tnnbtib.exe' has been instrumental in revealing a deep technical understanding of the malware's architecture and potential impact on infected systems. Analyzing the DLL imports with IDA Pro, particularly KERNEL32.DLL, USER32.DLL, WININET.DLL, ADVAPI32.DLL, CRTDLL.DLL, and wsock32.dll, exposed the malware's extensive integration into critical system processes and network functionalities. These dependencies suggest the malware's capabilities for system manipulation, such as unauthorized process control and user interface interaction, key for stealth and evasion. Furthermore, the reliance on network-related DLLs indicates sophisticated mechanisms for data exfiltration and command and control operations, highlighting the

malware's potential for continuous exploitation and targeted attacks.

The incorporation of ADVAPI32.DLL and CRTDLL.DLL implies a deeper system barricade, allowing the malware to manipulate system configurations, and registry keys, and manage services. This level of access suggests the malware is designed not just to perform malicious activities but to persist undetected within the host system. The technical insights gained from unpacking and analyzing 'tnnbtib.exe' underscore its capability to cause significant disruption and surveillance on infected systems, emphasizing the necessity for advanced detection and mitigation strategies against such sophisticated cyber threats.

VII. Conclusions from Dynamic and Static Analysis

The dynamic analysis of 'tnnbtib.exe' provided a comprehensive view of its operational behavior and potential impacts. The malware's sophisticated evasion techniques, particularly its use of the Run registry key for persistence, highlight its advanced capabilities in avoiding detection and maintaining a presence on the infected system. These findings, combined with the static analysis, contribute significantly to our understanding of malware's full spectrum of capabilities and intentions.

The dynamic analysis of 'tnnbtib.exe' has yielded insightful conclusions about its behavior and potential impacts on infected systems. Four primary indicators of compromise emerged as significant:

- File Creation: A critical indicator was the creation of the 'tnnbtib.exe' file within the C:\Windows directory. This action not only signifies the malware's attempt to embed itself into the system but also serves as a red flag for unauthorized file creation in a sensitive system directory, a common tactic for evasion and persistence.
- Network Communication: The malware engaged actively in network communication, utilizing DNS and HTTP protocols. These communications were directed towards various external IP addresses and domains, indicating potential command and control activities and data exfiltration attempts. This network behavior is a key indicator of the malware's capability to interact with remote servers, potentially receiving instructions or transmitting collected data.
- Process Creation: The initiation of a process with the same 'tnnbtib.exe' executable file was observed. This process creation is indicative of the malware's operational execution and its interaction with the system's resources. The repetitive spawning of its process highlights the malware's efforts to maintain active operation or reinstate itself after interruption.
- Registry Alteration: The malware made significant alterations to the system registry, particularly modifying keys associated with system startup and internet settings. These changes, especially in the HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Run key, are indicative of the malware's attempt to achieve persistence and potentially modify system behavior to suit its objectives.

In conclusion, these indicators of compromise - file creation, network communication, process creation, and registry alteration - when viewed in conjunction with the static analysis findings, provide a comprehensive understanding of 'tnnbtib.exe's' capabilities and intentions. They underline the malware's sophistication in terms of evasion, persistence, and operational execution, contributing significantly to the overall assessment of the threat it poses.

Acknowledgments

I would like to extend my sincere gratitude to Professor Joseph Opacki for his invaluable guidance and expertise in the field of Malware Reverse Engineering. His extensive knowledge and hands-on teaching approach have been instrumental in deepening my understanding of the complexities involved in analyzing and mitigating malware threats. Professor Opacki's dedication to the subject and commitment to educating his students have significantly contributed to the success of this research and my personal growth as a cybersecurity professional.

REFERENCES

[1] [1] M. K. A., "Learning Malware Analysis," in Learning Malware Analysis, 1st ed. San Francisco, CA, USA: No Starch Press, 2018, pp. 175-179.

[2] Microsoft, "Use the C run-time," Microsoft Docs. [Online]. Available: https://learn.microsoft.com/en-us/troubleshoot/developer/visualstudio/cpp/libraries/use-c-run-time. [Accessed: 24-11-2023]

[3] VirusTotal, "Behavior report for file e1a4b00d24aeb67fec1341103f94e1256d4cb5a696dd02847d9eadb60be4ec55," VirusTotal. [Online]. Available: https://www.virustotal.com/gui/file/e1a4b00d24aeb67fec1341103f94e1256d4cb5a696dd02847d9eadb60be4ec55/behavior. [Accessed: 23-11-2023]

[4] Opacki, J., "DFOR-761-001 Malware Reverse Engineering," in George Mason University, Fairfax, VA, USA, November 2023. [Week 12 Lecture].

[5] MITRE, "MITRE ATT&CK," MITRE ATT&CK. [Online]. Available: https://attack.mitre.org/. [Accessed: 11-23-2023]