# Dissecting PipeMagic: Inside the architecture of a modular backdoor framework

Among the plethora of advanced attacker tools that exemplify how threat actors continuously evolve their tactics, techniques, and procedures (TTPs) to evade detection and maximize impact, PipeMagic, a highly modular backdoor used by Storm-2460 masquerading as a legitimate open-source ChatGPT Desktop Application , stands out as particularly advanced. Beneath its disguise, PipeMagic is a sophisticated malware framework designed for flexibility and persistence. Once deployed, it can dynamically execute payloads while maintaining robust command-and-control ( C2) communication via a dedicated networking module. As the malware receives and loads payload modules from C2, it grants the threat actor granular control over code execution on the compromised host. By offloading network communication and backdoor tasks to discrete modules, PipeMagic maintains a modular, stealthy, and highly extensible architecture, making detection and analysis significantly challenging. Microsoft Threat Intelligence encountered PipeMagic as part of research on an attack chain involving the exploitation of CVE-2025-29824 , an elevation of privilege vulnerability in Windows Common Log File System (CLFS). We attributed PipeMagic to the financially motivated threat actor Storm-2460, who leveraged the backdoor in targeted attacks to exploit this zero-day vulnerability and deploy ransomware. The observed targets of Storm-2460 span multiple sectors and geographies, including the information technology (IT), financial, and real estate sectors in the United States, Europe, South America, and Middle East. While the impacted organizations remain limited, the use of a zero-day exploit, paired with a sophisticated modular backdoor for ransomware deployment, makes this threat particularly notable. This blog provides a comprehensive technical deep dive that adds to public reporting, including by ESET Research and Kaspersky. Our analysis reveals the wide-ranging scope of PipeMagic s internal architecture, modular payload delivery and execution mechanisms, and encrypted inter-process communication via named pipes. The blog aims to equip defenders and incident responders with the knowledge needed to detect, analyze, and respond to this threat with confidence. As malware continues to evolve and become more sophisticated, we believe that understanding threats such as PipeMagic is essential for building resilient defenses for any organization. By exposing the inner workings of this malware, we also aim to disrupt adversary tooling and increase the operational cost for the threat actor, making it more difficult and expensive for them to sustain their campaigns. PipeMagic: Technical analysis PipeMagic has been used by Storm-2460 in multiple instances as part of pre-exploitation activity for attack chains involving CVE-2025-29824. Microsoft Threat Intelligence observed Storm-2460 using the

certutil

utility to download a file from a legitimate website that was previously compromised to host the threat actor s malware. The downloaded payload is a malicious

MSBuild

file that ultimately drops and executes PipeMagic in memory. Once PipeMagic is running, the threat actor performs the CLFS exploit to escalate privileges before launching their ransomware. The first stage of the PipeMagic infection execution begins with a malicious in-memory dropper disguised as the open-source ChatGPT Desktop Application project. The threat actor uses a modified version of the GitHub project that includes malicious code to decrypt and launch an embedded payload in memory. The embedded payload is the PipeMagic malware, a modular backdoor that communicates with its C2 server over TCP. Once active, PipeMagic receives payload modules through a named pipe and its

C2 server. The malware self-updates by storing these modules in memory using a series of doubly linked lists. These lists serve distinct purposes for staging, execution, and communication, enabling the
threat actor to interact and manage the backdoor s capabilities throughout its lifecycle. Internal linked list structures In our analysis, we identified the use of four distinct doubly linked list structures, each serving a unique function within the backdoor s architecture:
Payload linked list: Stores raw
payload modules in each node, representing the initial stage of modular deployment. Execute linked list: Contains
payload modules that have been successfully loaded into memory and are ready for execution.
Network linked list: Contains networking modules responsible for
C2 communication. Unknown linked list: This structure lacks an immediately observable function. Based on behavioral analysis, we hypothesize it is leveraged dynamically by loaded payloads rather than the core backdoor logic itself. In the next sections, we will detail how each of these linked lists is populated and utilized as we walk through the malware s execution flow and capabilities. Populating the
payload linked list The malware uses a doubly linked list structure to manage its
payload modules, with each node encapsulating a
payload in its raw Windows Portable
Executable (PE) format. Before initializing this list, the malware generates a 16-byte random bot identifier unique to the infected host.
Figure 1. Bot ID generation It then
spawns a dedicated thread to establish a named
pipe for
payload delivery. The
pipe is created using the format \\.\
pipe\1. , where the bot ID is the randomly generated ID above. Figure 2.
Pipe name generation A bidirectional named
pipe is established, enabling both read and write operations between the malware (acting as the
pipe client) and the
payload delivery mechanism (
pipe server). The malware continuously listens on this
pipe, reading incoming
payload modules in a loop. For each module, the malware reads the

payload s length from the
pipe, allocates memory accordingly, reads the
payload content, and adds it to the
payload module linked list. Figure 3. Connecting and reading
pipe data The structure below represents the layout of the
pipe data being delivered to the malware from the
pipe server. struct
pipe_data_struct { DWORD module_setup_flag;
// add module node (1) or stop reading
pipe (2) DWORD module_index;
// module index DWORD module_name;
// module name DWORD module_body_len;
// length of module data DWORD module_body_SHA1_hash;
// SHA1 hash of module data BYTE module_body[];
// pointer to module data }; After the
pipe data is read, the malware extracts the module body and decrypts it
using RC4 with the following hardcoded 32-byte key: 00000000 7b c6 ea 4b 9d
82 ec d5 fb 31 05 87 b9 8c be 3b
|{ K.. 1.. . ;
| 00000010 b8 f7 c9 f7 29 fa 9e 87 27 41 a9 e3 be 34 4d fa
| ) ..'A 4M
| The malware then computes the SHA-1 hash of the decrypted data and
compares it against the hash provided in the
pipe data to verify integrity. Figure 4. Decrypting module data and
performing hash validation Upon successful validation, the malware
constructs the following node structure representing the
payload module and inserts it
at
the head of the
payload linked list. This same structure is also used later in the execute
linked list. struct __declspec(align(8)) module_node { module_node *next;
// next node module_node *prev;
// previous node DWORD module_index;
// module index DWORD exec_ll_module_index;
// module index in the execute linked list BYTE *module_data_ptr;
// module pointer DWORD module_data_len;
// module length DWORD module_name;
// module name int module_entry;
// module entrypoint int module_attribute;

// attribute (4: aPLib compressed, 8: RC4 encrypted, 12: both) BYTE module_initialized_flag;
// initialized flag BYTE *module_hash_ptr;
// module SHA1 hash DWORD module_hash_len;
// module SHA1 hash length }; Figure 5. Populating payload module with pipe data The malware communicates the result of this operation back to the pipe server using the following response codes: Code Description 0x0 Success module node created and inserted 0x1 Invalid pipe data size 0x3 Failed to create a payload module node 0xA SHA-1 hashing of module data failed 0xB Hash mismatch integrity check failed This thread remains active throughout the backdoor s lifecycle, allowing the threat actor to continuously deliver new payloads through the named pipe. The thread only terminates when the malware receives a module setup flag value of 2 in the pipe data, signaling the end of payload delivery. Malware configuration The malware uses a well-defined configuration structure to manage its operational parameters. The outermost configuration is represented by the following structure. It consists of a length field followed by a data buffer of that length: struct backdoor_config { DWORD config_len; BYTE config_data[config_len]; } If the config_len field is the constant 0x5A , the hardcoded configuration is deemed invalid, and the malware simply operates in local execution mode, communicating exclusively with the loopback interface at 127.0.0 [.]1:8082 . This mode is likely used for testing or staging purposes, allowing the malware to simulate C2 interactions without external network dependencies. The config_data field itself contains multiple configuration blocks. Each block follows a consistent internal format: struct config_block { DWORD block_index; DWORD block_data_len; BYTE block_data[block_data_len]; } The malware uses the block_index field to identify and retrieve specific configuration blocks as needed. Below is a breakdown of the known block indices and their corresponding data: Block index Block description Block data 1 C2 config block aaaaabbbbbbb.eastus.cloudapp.

azure[.]com:443 2 Unknown 43 3 Backdoor s max up time 172800 4 Unknown 120 It s currently unclear how blocks with indices 2 and 4 are used. These values do not appear to influence the malware s core functionality. However, they are transmitted to the
C2 server alongside system information during the initial connection. The data in block index 1 is itself another configuration block. It contains the actual
C2 address used by the malware, which is aaaaabbbbbbb.eastus[.]cloudapp. azure[.]com:443 . This domain has been disabled by Microsoft . Figure 6. Extracting configuration Launching networking module The backdoor does not communicate with
C2 directly. Instead, it delegates this task to a
network module in the
network linked list. First, it populates the
network linked list with module nodes. Each node contains an
executable module responsible for handling
C2 communication. In the sample analyzed, the
network module data is embedded within the backdoor binary. This data is first
XOR
-decrypted using the following hardcoded 32-byte key, then decompressed using the aPLib compression algorithm. 00000000 91 df 5d 0e 9c 64 cd bd c2 46 f2 4b 6b ce 4a dc
|. ]..d F Kk J
| 00000010 aa 38 f9 60 0f e4 e4 98 ed 05 46 f1 ca d9 54 c5
| 8 `. . .F T
| Figure 7. Decrypting
network module data Using the decrypted module data, the malware populates the following structure representing a module node in the
network linked list. struct
network_module_node { __int64 module_index;
// module index in
network linked list BYTE *module_base;
// pointer to module base __int64 module_size;
// module size __int64 module_main_func;
// pointer to the main function BYTE *module_entrypoint;
// pointer to the module's entry point BYTE terminate_flag;
// terminate flag }; Once the node is initialized and the module is loaded into memory, the malware executes the module s entry point, passing a

pointer to its own main function as a parameter. Figure 8. Launching network module s entry point In the network module s entry point, the module sets its third argument to its actual main function. This allows the backdoor to assign the module s main function to the module_main_func field in the node structure, allowing the backdoor to call this function directly. Figure 9.
Network module s entry point Finally, the backdoor inserts the module node into the network linked list and invokes its main function, passing the C2 address extracted from the configuration. Figure 10. Launching network module s main function
Network module: Establishing C2 connection When launched by the backdoor, the network module first exports and registers three of its internal functions for use by the backdoor: A function to send data to the C2 server over TCP A function that returns the constant value 0x8ca A function to set a stop signal, instructing both the backdoor and the network module to terminate all C2 communications The backdoor uses the first exported function to send data to the C2 server through the network module, rather than handling communication directly. Figure 11.
Network module s exported functions After initialization, the network module begins its communication routine with the C2 server. On each execution, it limits itself to a maximum of five communication attempts with the C2. Once a TCP connection is established, the module sends the following HTTP GET request to initiate communication with the C2 server. The path includes a randomly generated 16-character hexadecimal string that is unique for each connection. GET / HTTP/1.1 Host: aaaaabbbbbbb.eastus.cloudapp.
azure[.]com Connection: Upgrade Pragma: no-cache Cache-Control: no-cache User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.110 Safari/537.36 Upgrade: websocket Origin: aaaaabbbbbbb.eastus.cloudapp.

azure[.]com Sec-WebSocket-Version: 13 Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7 Sec-WebSocket-Key:
4nnwIaDMxE5LZ6iNQ4XE3w
== Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits
Figure 12. Setting up and sending initial GET request Once a valid response is received from the
C2 server, the
network module transfers execution back to the backdoor.
At
this point, the backdoor collects system information and sends it to the
C2 server using the
network module s communication function (annotated as
C2_send_request in Figure 11). System information collection After the
C2 connection is successfully established by the
network module, the backdoor collects a comprehensive set of system and internal state information to send back to the
C2 server: Generated bot ID
Network module s index in the
network linked list Operating system version Computer name Malware executable name Malware process ID Whether the host belongs to the Network Configuration Operators SID group Domain NetBIOS name Whether the malware is running as a 64-bit process List of all LAN domain groups the host belongs to Integrity level of the malware process User domain name Session ID of the malware process Host s IP address Malware s current working directory Data from all nodes in the execute linked list Data from all nodes in the unknown linked list This host information is commonly collected by backdoors to be used as the host s unique identifier when the malware attempts to establish a connection with its
C2 server. Once this information is gathered, the PipeMagic backdoor invokes the
network module s communication function to transmit the data to the
C2 server over the established TCP socket. After the data is sent, execution is handed back to the
network module, which waits for and receives the
C2 response. Finally, the
network module transfers
control
back to the backdoor, passing along the

C2 response so the backdoor can proceed with executing its core malicious capabilities. Processing
C2 response Once the backdoor receives a response from the
C2 server, it parses the data to extract the outer processing command. This command determines how the backdoor should handle the response and what actions to take next. Below is a list of known processing codes and their corresponding functionalities: Processing code Processing data
Functionality 0x1 Backdoor code and data Executes core backdoor functionality using modules from the execute and
payload linked lists 0x3 Module index Looks up a module node with the provided index and execute the module code 0x5 A message Sends the received message back to the
C2 server as an acknowledgment or
echo 0x7 N/A Shuts down the
network module and stops all
C2 communication 0x8 Backdoor code and data Executes backdoor functionality using modules from the unknown linked list 0xA Module node argument Invokes all modules in the execute linked list with the specified argument Backdoor capabilities: Execute and
payload linked list Among all the outer processing commands, processing code 0x1 is the most significant. When this code is received, the associated processing data contains inner backdoor commands and arguments that enable PipeMagic to perform a wide range of backdoor operations. Below is a list of known backdoor codes and their corresponding functionalities: Backdoor code Backdoor arguments Functionality 0x1 N/A Retrieves metadata from all module nodes in the
payload linked list 0x2 arg1: Module index arg2: Module data length arg3: Module name arg4: Module attribute arg5: Module SHA1 hash Inserts a new module node into the
payload linked list and initializes it with the provided data; Skips insertion if a matching module (by index and hash) already exists 0x3 arg1: Module index arg2: Hash flag arg3: Write offset arg4: Write length arg5: Payload data Locates a module node in the
payload linked list using the provided index and writes data
at
the specified offset; if the hash flag is provided, recomputes and updates the SHA-1 hash after RC4 encryption and aPLib compression (depending on the module s attribute) 0x4 arg1: Module index arg2: Read offset arg3: Read length Reads a segment of data from a module node in the

payload linked list 0x5 arg1: Module index Deletes a module node from the
payload linked list 0x6 arg1: Module index arg2: Write offset arg3:
Payload data arg4: Write length Writes data to a module node without
updating the SHA-1 hash 0x7 arg1: Module index Retrieves the SHA-1 hash of a
module node in the
payload linked list 0x9 N/A Retrieves data from all module nodes in the
execute linked list 0xA arg1: Module index Retrieves data from a specific
module node in the execute linked list 0xB arg1:
Payload module index arg2: Execute module index arg3: Initialization flag
Loads a
payload module into memory and binds it to a node in the execute linked
list, then invokes its entry point 0xC arg1: Module index Executes the entry
point of a module node in the execute linked list 0xD N/A Retrieves the user
s domain name 0xE N/A Retrieves the current
C2 processing code and data 0xF N/A Renames the malware
executable to :fuckit and marks it for self-deletion 0x10 arg1: Lower index
arg2: Upper index Deletes all module nodes in the
payload linked list within the specified index range 0x11 arg1: Module name
Deletes a module node in the
payload linked list by name instead of index 0x13 N/A Enumerates all running
processes and collects session ID, PID, PPID, creation time,
executable path, user domain, and architecture (32-bit or 64-bit) 0x14
arg1: Module index arg2: New module name arg3: Module hash length arg4:
Module hash arg5:
Pipe data to send arg6:
Pipe name arg7: Max elapsed time Replaces a module node in the
payload linked list; sends data to a named
pipe and parses the response to receive the
payload module data 0x15 arg1: Module index arg2: New module name arg3: New
module attribute arg4: Module hash length arg5: Module hash arg6: Module
data length arg7: Module data Replaces a module node in the
payload linked list with a new one; the provided data is RC4-decrypted,
aPLib-decompressed, and validated by SHA-1 hash before being added to the
payload module node 0x16 N/A Recollects system information (same as the
initial
C2 handshake) 0x17 arg1: Module index arg2:
Pipe data 1 arg3:
Pipe data 2 arg4: Max elapsed time arg5:
Pipe name Extracts and RC4-encrypts data from a module in the

payload linked list; sends it to a named pipe along with the provided pipe data. Backdoor results are delivered to C2 over TCP. These inner backdoor codes provide the threat actor with granular control over module management, execution, and system reconnaissance, making PipeMagic a highly modular and extensible backdoor. Backdoor capabilities: Unknown linked list Processing code 0x8 functions similarly to processing code 0x1 in that it also contains inner backdoor code and data. However, this command is specifically designed to interact with the unknown linked list. The purpose of this linked list remains unclear. It does not appear to play a critical role in the malware s core functionality on the infected system. Below is a list of known backdoor codes associated with this processing command and their corresponding functionalities: Backdoor code Backdoor arguments Functionality 0x1 N/A Retrieves metadata from all module nodes in the unknown linked list 0x2 arg1: Module index Looks up a module node in the unknown linked list and extract its data 0x3 arg1: Module index Deletes a module node from the unknown linked list using the specified index 0x7 arg1: Module index arg2: New module size Resizes the data buffer of a module node in the unknown linked list, either expanding or shrinking it based on the provided size While the exact role of this list remains unclear, its structure and command handling mirror those of the payload and execute linked lists, suggesting it may serve as a staging area or auxiliary buffer for dynamically loaded modules. Mitigation and protection guidance Microsoft recommends the following mitigations to reduce the impact of activity associated with PipeMagic and Storm-2460: Ensure that tamper protection is enabled in Microsoft Defender for Endpoint. Enable network protection in Microsoft Defender for Endpoint. Run endpoint detection and response (EDR) in block mode so that Microsoft Defender for Endpoint can block malicious artifacts, even when your non-Microsoft antivirus does not detect the threat or when Microsoft Defender Antivirus is running in passive mode. EDR in block mode works behind the scenes to remediate malicious artifacts that are detected post-breach. Configure investigation and remediation in full automated mode to let Microsoft Defender for Endpoint take immediate action on alerts to resolve breaches, significantly reducing alert volume. Use Microsoft

Defender Vulnerability Management to assess your current status and deploy any updates that might have been missed. Turn on
cloud-delivered protection in Microsoft Defender Antivirus or the equivalent for your antivirus product to cover rapidly evolving attacker tools and techniques.
Cloud-based machine learning protections block a majority of new and unknown variants. Microsoft Defender XDR
detections Microsoft Defender XDR customers can refer to the list of applicable
detections below. Microsoft Defender XDR coordinates
detection, prevention, investigation, and response across endpoints, identities, email, apps to provide integrated protection against attacks like the threat discussed in this blog. Customers with provisioned access can also use Microsoft Security Copilot in Microsoft Defender to investigate and respond to incidents,
hunt for threats, and protect their organization with relevant threat intelligence. Microsoft Defender Antivirus Microsoft Defender Antivirus detects this threat as the following malware: PipeMagic (Win32/64) Microsoft Defender for Endpoint The following Microsoft Defender for Endpoint alerts can indicate associated threat activity: PipeMagic malware was detected PipeMagic malware was prevented An active PipeMagic malware was blocked An active PipeMagic malware process was detected while executing and terminated The following alerts might also indicate threat activity related to this threat. Note, however, that these alerts can be also triggered by unrelated threat activity. A file or
network connection related to a
ransomware-linked emerging threat activity group detected Microsoft Defender Vulnerability Management Microsoft Defender Vulnerability Management surfaces devices that may be affected by the following vulnerabilities used in this threat: CVE-2025-29824 Microsoft Security Copilot Security Copilot customers can use the standalone experience to create their own prompts or run the following pre-built promptbooks to automate
incident response or investigation tasks related to this threat:
Incident investigation Microsoft User analysis
Threat actor profile Threat Intelligence 360 report based on MDTI article Vulnerability impact assessment Note that some promptbooks require access to plugins for Microsoft products such as Microsoft Defender XDR or Microsoft Sentinel. Threat intelligence reports Microsoft customers can use

the following reports in Microsoft products to get the most up-to-date information about the
threat actor, malicious activity, and techniques discussed in this blog. These reports provide the intelligence, protection information, and recommended actions to prevent, mitigate, or respond to associated threats found in customer environments. Microsoft Defender XDR Threat analytics Vulnerability Profile: CVE-2025-29824 Windows Common Log File System Microsoft Security Copilot customers can also use the Microsoft Security Copilot integration in Microsoft Defender Threat Intelligence, either in the Security Copilot standalone portal or in the embedded experience in the Microsoft Defender portal to get more information about this
threat actor. Indicators of
compromise
Indicator Type Description aaaaabbbbbbb.eastus.cloudapp.
azure[.]com:443 Domain PipeMagic s
C2 domain dc54117b965674bad3d7cd203ecf5e7fc822423a3f692895cf5e96e83fb88f6a File SHA-256 hash In-memory dropper (trojanized ChatGPT desktop application)
4843429e2e8871847bc1e97a0f12fa1f4166baa4735dff585cb3b4736e3fe49e File SHA-256 hash PipeMagic backdoor (unpacked in memory)
297ea881aa2b39461997baf75d83b390f2c36a9a0a4815c81b5cf8be42840fd1 File SHA-256 hash PipeMagic
network module (unpacked in memory) References https:
//www.kaspersky.com/about/press-releases/kaspersky-uncovers-pipemagic-backdoor-attacks-businesses-through-fake-chatgpt-application https:
//x.com/ESETresearch/status/1899508656258875756
Learn more For the latest security research from the Microsoft Threat Intelligence community, check out the Microsoft Threat Intelligence Blog: https:
//aka.ms/threatintelblog . To get notified about new publications and to join discussions on social media, follow us on LinkedIn
at
https:
//www.linkedin.com/showcase/microsoft-threat-intelligence , and on X (formerly Twitter)
at
https:
//x.com/MsftSecIntel . To hear stories and insights from the Microsoft Threat Intelligence community about the ever-evolving threat landscape,

listen to the Microsoft Threat Intelligence podcast: https://thecyberwire.com/podcasts/microsoft-threat-intelligence . The post Dissecting PipeMagic: Inside the architecture of a modular backdoor framework appeared first on Microsoft Security Blog .