Cyberespionage Group Earth Hundun's Continuous Refinement of Waterbear and Deuterbear

By By: Cyris Tseng, Pierre Lee Apr 11, 2024 Read time: 13 min (3461 words)

Published: 2024-04-11 · Archived: 2026-04-05 15:15:30 UTC

Malware

Our blog entry provides an in-depth analysis of Earth Hundun's Waterbear and Deuterbear malware.

Summary

- Earth Hundun is a cyberespionage-motivated threat actor that has been active for several years in the Asia-Pacific region, targeting the technology and government sectors.
- The group has been known for employing several tools and techniques, including Waterbear, a malware entity that has had over 10 versions since 2009.
- Waterbear is known for its complexity, as it uses a number of evasion mechanisms to minimize the chance of detection and analysis. Succeeding versions have added enhancements that make it even more troublesome to deal with.
- In 2022, Earth Hundun began using the latest version of Waterbear — also known as Deuterbear — which has several changes, including anti-memory scanning and decryption routines, that make us consider it a different malware entity from the original Waterbear.
- Our blog entry provides an in-depth analysis of these two malware types in Earth Hundun’s bag of tools.

Introduction

We recently observed a surge in cyberattacks targeting a number of organizations in various sectors such as technology, research, and government. These attacks involve a malware family known as Waterbear that is linked to the cyberespionage group Earth Hundun (also known as BlackTech), a threat actor that focuses on gathering intelligence from technology and government organizations, particularly in the Asia-Pacific region.

Among the group’s arsenal of weapons, the Waterbear backdoor is one of the most complex, with a wide array of anti-debug, anti-sandbox, and general antivirus-hindering techniques. Moreover, the frequent updates from its developers have led to even more evasion tactics, including enhancements of its loader, downloader, and communication protocol. This report will delve into the latest techniques Earth Hundun has implemented with Waterbear and provide an analysis of its latest iteration, Deuterbear.

Waterbear details

Waterbear has had over 10 versions since 2009, with the version number directly visible in the configuration. Despite available solutions for older versions, its operators typically persist in enhancing infection flows until a

successful compromise. Therefore, it is common to find multiple versions coexisting within the same timeframe and even within the environments of the same victims.

Interestingly, some Waterbear downloaders have been seen using command-and-control (C&C) servers with internal IP addresses (for instance, the downloader with hash *6b9a14d4d9230e038ffd9e1f5fd0d3065ff0a78b52ab338644462864740c2241* uses the internal IP 192.168.11[.]2 as its C&C server).

This suggests that the attackers might have in-depth knowledge of their victims' networks, employing multilayered jump servers to evade detection. Such tactics underscore the sophisticated nature of these attacks, which are designed to stealthily maintain presence and control within compromised environments.

Attack chain and TTPs of Waterbear

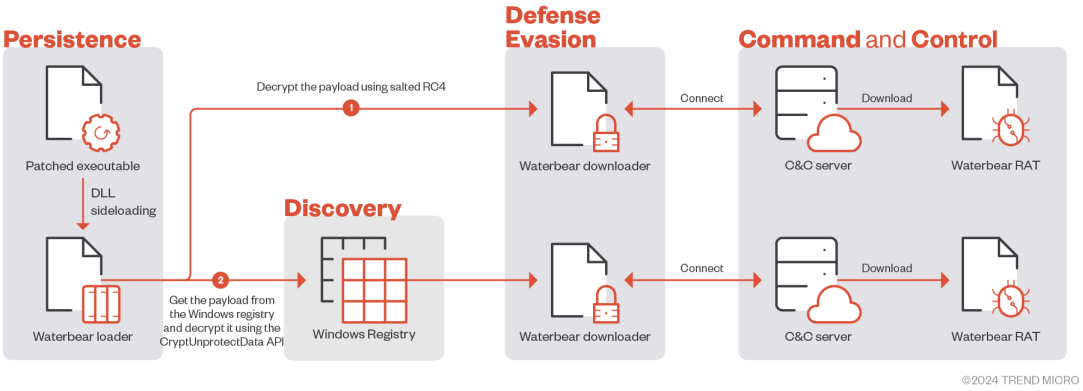


Figure 1. Waterbear infection flow chart

For the launcher, Waterbear uses a legitimate executable to load its custom DLL file. In some cases, its operators patched the legitimate executable to modify the import table. This includes adding the DLL with the same file name at ordinal 0, enabling a smooth launch of the loader via DLL sideloading. This strategy allows Earth Hundun to run its custom DLL loader and avoid detection.

Figure 2. Modifying the import table with a legitimate executable

Loader

Based on the diagram shown in Figure 1, there are two decryption routines used by Waterbear to decrypt the encrypted downloader.

We observed that recent Waterbear loader routines commonly use the same custom salted RC4 decryption, accompanied by a similar obfuscation pattern, to decrypt the downloader. This approach is consistent across downloader versions 0.13, 0.16, and 0.24. In contrast, earlier versions of the Waterbear loader were barely obfuscated, if at all.

```
result = a1;
v16 = 256;
do
{
  v10 = (*result + result[v13 - a1] + v10) % 256;
  v15 = *result;
  *result++ = *(a1 + v10);
  *(a1 + v10) = v15;
  --v16;
}
while ( v16 );
```

```
v27 = a1;
v28 = 64i64;
do
{
  v29 = *v27;
  v30 = v27[&v60 - a1];
  v27 += 4;
  v31 = (((v29 + v11 + v30) >> 31) + v29 + v11 + v30) - ((v29 + v11 + v30) >> 31);
  v32 = v31 + v27[&v61 - a1 - 4];
  *(v27 - 4) = a1[v31];
  a1[v31] = v29;
  v33 = *(v27 - 3);
  v34 = (((v33 + v32) >> 31) + v33 + v32) - ((v33 + v32) >> 31);
  v35 = v34 + v27[&v62 - a1 - 4];
  *(v27 - 3) = a1[v34];
  a1[v34] = v33;
  v36 = *(v27 - 2);
  v37 = (((v36 + v35) >> 31) + v36 + v35) - ((v36 + v35) >> 31);
  v38 = v37 + v27[v63 - a1 - 4];
  *(v27 - 2) = a1[v37];
  a1[v37] = v36;
  v39 = *(v27 - 1);
  result = (((v39 + v38) >> 31) + v39 + v38) - ((v39 + v38) >> 31);
  --v28;
  v11 = result;
  *(v27 - 1) = a1[result];
  a1[result] = v39;
}
while ( v28 );
```

Figure 3. Past Waterbear variants did not use obfuscation in the RC4 KSA stage (top) compared to more recent variants that use obfuscation (bottom)

```
do
{
  v4 = (v4 + 1) % 256;
  v8 = *(v4 + a1);
  v15 = (v15 + v8) % 256;
  v9 = v15;
  *(v4 + a1) = *(v15 + a1);
  *(v9 + a1) = v8;
  v10 = *((*(v4 + a1) + v8) % 256 + a1);
  if ( v7 % 2 )
    v11 = ~v10;
  else
    v11 = v7 ^ v10;
  v12 = (v7 + a4);
  v13 = v11 ^ *(v17 + v7 + a4);
  ++v7;
  *v12 = v13;
}
while ( v7 < a3 );
```

```
do
{
  v4 = (((v4 + 1) >> 31) + v4 + 1) - ((v4 + 1) >> 31);
  v34 = v32[v4];
  v17 = (((v17 + v34) >> 31) + v17 + v34) - ((v17 + v34) >> 31);
  v32[v4] = v32[v17];
  v32[v17] = v34;
  v35 = v34 + v32[v4];
  v36 = v32[(BYTE4(v35) + v34 + v32[v4]) - BYTE4(v35)];
  if ( ((v31 >> 31) ^ v31 & 1) == v31 >> 31 )
    v37 = v31 ^ v36;
  else
    v37 = ~v36;
  v38 = a4[v33];
  ++v31;
  *a4++ = v37 ^ v38;
}
while ( v31 < a3 );
```

Figure 4. Past Waterbear variants did not use obfuscation in the RC4 PRGA stage (top) compared to more recent variants that use obfuscation (bottom)

In some cases, Waterbear loaders routinely place the encrypted downloader in the registry in advance, with the downloader being decryptable only on the infected machine since it uses the CryptUnprotectData API. This

method is limited by the requirement that it must operate on the infected machine. However, it can prevent the victim from realizing that they are being attacked, while also hindering incident responders during investigation.

Downloader

Earth Hundun has been gradually refining its technique to bypass antivirus software adding a large amount of padding with 0x00 around the beginning and end to avoid detection. After decryption, the loader executes the shellcode directly and checks the debugger mode, initiating the Waterbear downloader.

1. Decrypts the function before using it and encrypts it again after use
2. After recovering the function address, they quickly move it to another place in memory and mess-up the original address.

For more detailed information, please refer to our previous report, specifically the section titled “Anti-memory scanning of shellcode payload.”

The configuration outlined in the previously mentioned report contains the information required for proper execution and communication with C&C server.

Data offset	**Data size**	**Data content**
0x00	0x10	Encryption/Decryption key for the functions
0x10	0x04	Remote access trojan (RAT) infection mark, which is also used for sleep time.
0x14	0x10	Version (such as 0.13, 0.16, 0.24, and so on)
0x24	0x0C	Mutex (not use for now)
0x34	0x78	C&C server address, which is XOR-encrypted with the key 0xFF; has each address with a maximum length of 0x28 and supports up to 3. If the downloader is intended to listen in on a specific port, this section will be filled with 0x00.
0xAC	0x02	Port number (might contain multiple numbers)
0xD8	0x10	traffic KEY_1, RC4 key of first traffic sent from victim
0xE8	0x10	traffic KEY_2, unique ID to identify victim
0xF8	0x10	traffic KEY_RANDOM (randomly generated by the downloader and the RC4 key of encrypted RAT sent from the C&C server)
0x108	0xC8	List of function addresses (for example, 0x8 * 25 functions)
0x1D0	0x64	List of function lengths (for example, 0x4 * 25 functions)
0x234	0x124	List of API addresses

0x358	0x90	List of encrypted API hash
0x3E8	0x78	List of library names

Table 1. The configuration structure of Waterbear downloader

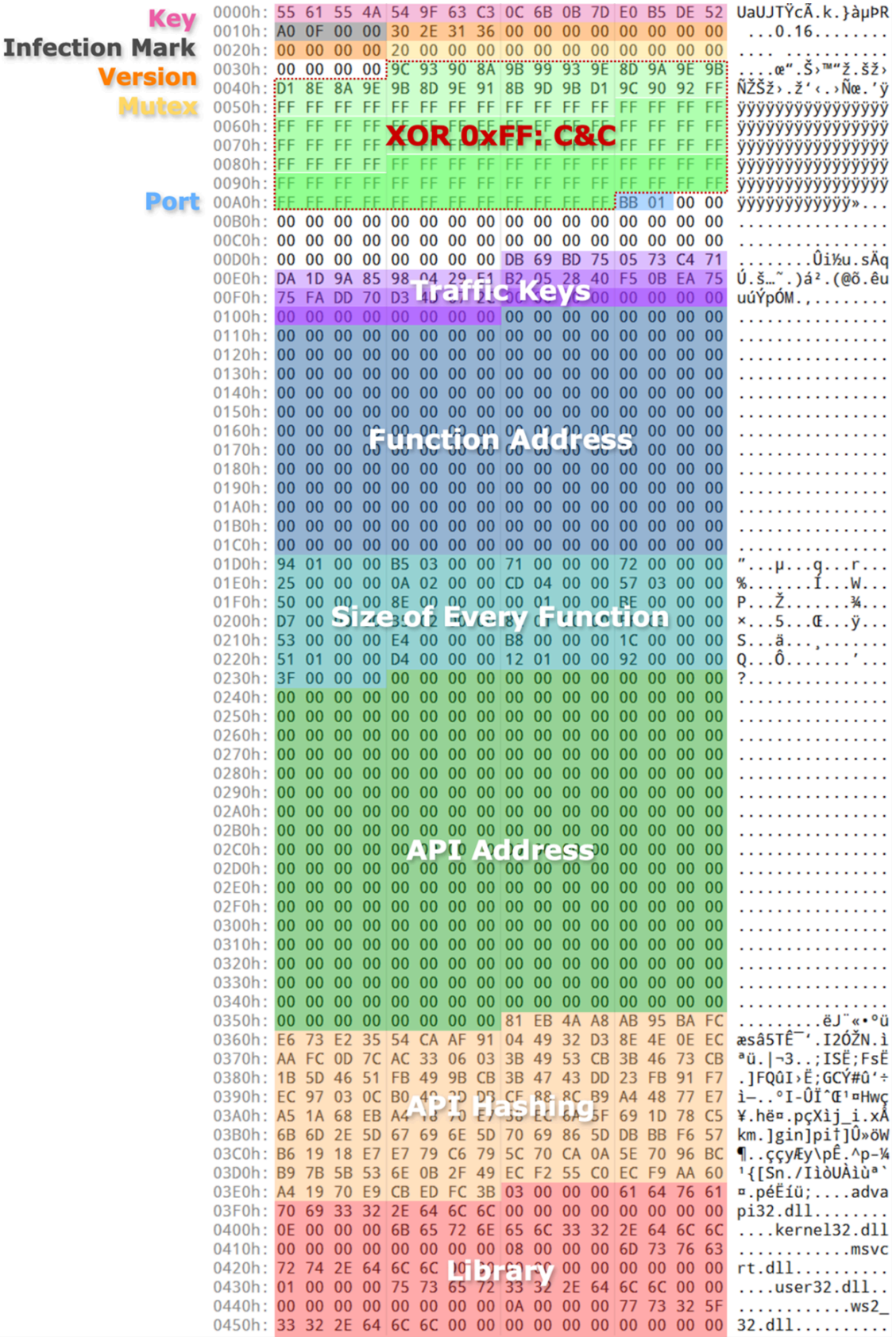


Figure 5. A screenshot showing the configuration structure of Waterbear downloader

For the network request, the downloader will set up the custom connection to deliver the next stage RAT as follows:

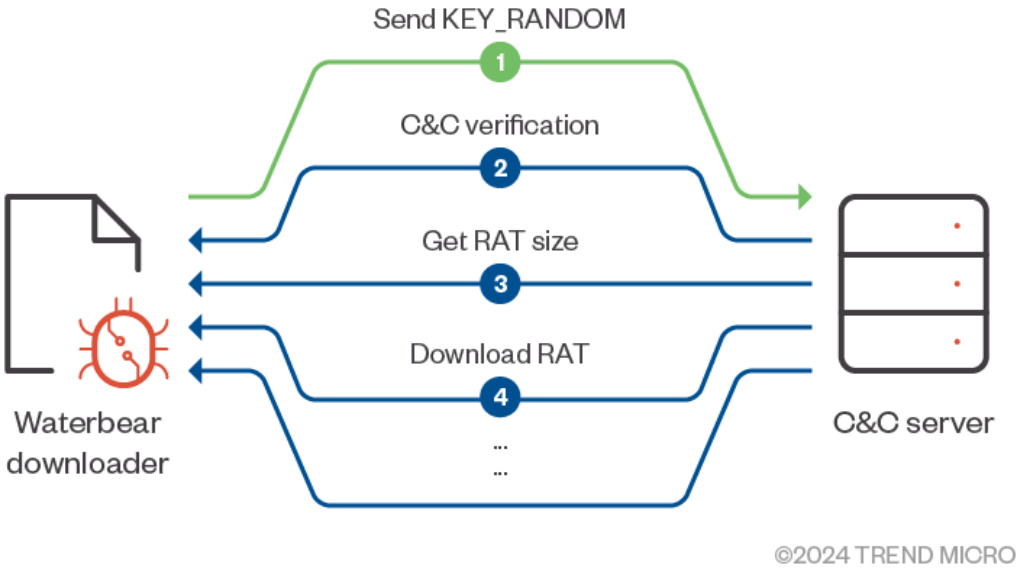


Figure 6. Network traffic to download the Waterbear RAT

Index	Direction	Encryption	Key
1	Victim -> C&C	Salted RC4 (10000 times)	KEY_1
2	C&C -> Victim	Salted RC4	KEY_RANDOM XOR reversed (KEY_1)
3	C&C -> Victim	Salted RC4	KEY_RANDOM
4	C&C -> Victim	Salted RC4	KEY_RANDOM

Table 2. Basic information about network traffic to download the Waterbear RAT

All of the packets have a 10-byte header with which to describe the information of data (keeping the same format as described in a report published by Palo Alto. However, the signature has been obfuscated over time by the threat actors to evade detection. The analysis of the latest protocol is shown here:

Send KEY_RANDOM

The downloader randomly generates the 16-byte key, *KEY_RANDOM*, and sends the packet to the C&C server with the format:

Offset	Size	Type	Content
0x00	0x10	Header	The 1st, 4th, and 6th are generated randomly and applied to encrypt other bytes in the header. 2nd: 0x40 XOR 6th byte 3rd: 0x1F XOR 1st byte 5th: 0x03 XOR 4th byte XOR ((1st byte >> 4) AND (6th byte << 4)) 7th: size_of_data XOR 1st byte

			8th: (size_of_data >> 8) XOR 6th byte 9th: (size_of_data >> 16) XOR 4th byte 10th: (size_of_data >> 24) XOR (4th byte << 4) AND (6th byte >> 4)
0x10	0x20	Data	0x00 – 0x10: <KEY_RANDOM> XOR “abcdefghijklmno\x00” 0x10 – 0x20: <KEY_RANDOM> XOR <KEY_2>

Table 3. Packet format for sending KEY_RANDOM.

The header contains the command code 0x40 0x1F, and the size of the data in the last four bytes by little-endian, but this variant’s obfuscation method is more complex than the previous version. The C&C server will perform the reversed calculation to decrypt the header and data while the *KEY_RANDOM* will be applied to the key of the salted RC4 in the next packets. The *KEY_2* is the unique ID to check the target.

C&C Verification

C&C server sends the packet to victim for verification with the format:

Offset	Size	Type	Content
0x00	0x10	Header	?? 40 1F ?? ?? ?? ?? ?? ?? ?? (The last 4 bytes are the size of the data with little-endian)
0x10	0x20	Data	The data contains the KEY_1, with the offset of KEY_1 being ((1st byte XOR 2nd byte) + 2)

Table 4. Packet format for C&C verification.

Get RAT Size

C&C server sends the packet for RAT size with the format:

Offset	Size	Type	Content
0x00	0x10	Header	?? 43 1F ?? 00 ?? 04 00 00 00
0x10	0x04	Data	The size of the RAT with little-endian.

Table 5. Packet format for getting the RAT size

Download RAT

C&C server sends the packet for RAT with the format:

Offset	Size	Type	Content

0x00	0x10	Header	?? 43 1F ?? 01 ?? ?? ?? ?? ?? (The last 4 bytes are size of data with little-endian)
0x10	Not Fixed	Data	The segment of next-stage RAT.

Table 6. Packet format for getting the RAT

This step repeatedly receives the packet from the C&C server until the whole RAT is delivered.

RAT command

Since TeamT5's article in 2020 discussing Waterbear's functions, there have been more of them that have been implemented, with the latest version shown in this table:

Command code (decimal)	**Capability**
2	Enumerate disk drives
3	List files
4	Upload file to C&C server
5	Download file from C&C server
6	Rename file
7	Create folder
8	Delete file
10	Execute file
11	Move file
12	Disguise meta data of file
13	File operation
806	Get system language, system time and Windows installation date
807	Enumerate Windows
809	Hide Windows
810	Show Windows
811	Close Windows

812	Minimize Windows
813	Maximize Windows
815	Screenshot
816	Set screenshot event signaled
817	Remote desktop
818	Enumerate process
819	Terminate process
821	Suspend process with pID
822	Resume process with pID
823	Get process module infomation
824	Get process module info (for file or object using the authenticode policy provider)
825	Get extended TCP table
826	SetTcpEntry Set state of the TCP connection with MIB_TCP_STATE_DELETE_TCB
827	Enumerate services
828 – 832	Manipulate service
833	Get C&C in downloader config
834	Set C&C in downloader config
1006	Start remote shell
1007	Exit remote shell
1008	Get PID of remote shell
1010	Download DLL and execute the export function “Start”
1300	Unknown
2011	Enumerate Registry
2012	Enumerate registry value
2013	Create registry key

2014	Set registry value
2015	Delete registry key
2016	Delete registry value
8001	Get current window
8004	Set the infection mark in registry HKCU\Console\Quick\Edit
8005	Terminate connection and RAT process
9010	Update C&C IP address
9011 -9018	Manipulate socket

Table 7. List of RAT command and corresponding functionalities.

For more details about Waterbear's past activities, please refer to our 2019 report.

Deuterbear details

The Deuterbear downloader, the latest Waterbear downloader, has been active since 2022 based on our telemetry. Because of significant updates in the decryption flow and configuration structure, we classify this variant as a distinct malware entity separate from the original Waterbear downloader category.

Attack chain and TTPs of Deuterbear

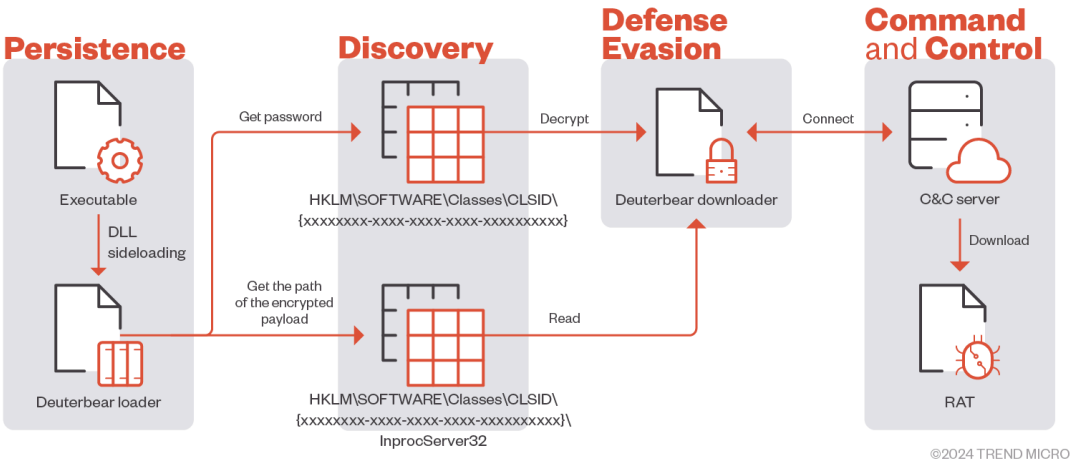


Figure 7. Deuterbear infection flow chart

Loader

The decryption flow is limited on the victim's side due to the API (CryptUnprotectData) and the need for more parameters, which are defined by the threat actor:

1. Query password from registry (HKLM|HKCU|HKCR)SOFTWARE\\Classes\\CLSID\\{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx} with key 'AppID'
2. Query path of encrypted downloader from registry (HKLM|HKCU|HKCR) SOFTWARE\\Classes\\CLSID\\{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx}\\InprocServer32
3. Downloader decryption

    a. XOR with password from offset 16~999

    b. *CryptUnprotectData* without password

    c. XOR with password from offset 0~999

    d. *CryptUnprotectData* with password

Note that the CLSID value is unique and defined during malware installation.

Downloader

The Deuterbear downloader **enables HTTPS tunnel** to protect the network traffic and implements the following obfuscation methods for anti-analysis:

1. Breaking the function using jmp
2. Checking debugger mode by process time
3. Checking sandbox environment by API, Sleep, which is normal operation
4. Checking execution in specific time, like 9~10 o'clock
5. Implementing anti-memory scanning

Anti-memory scanning, which is inherited from the Waterbear downloader, encrypts all function blocks (except for the function involving decode routine) with a fixed key defined in configuration. However, the Deuterbear downloader executes the desired function in new virtual memory, and not in the local address that stores all the encrypted function blocks.

```
v28 = *(_QWORD *)(HeaderStructure + 642);// function offset: 5a70
v29 = *(unsigned __int16 *)(HeaderStructure + 936);// function length: BD
RunEncryptCode(v24, HeaderStructure + 1600, 77i64, 1i64, 0i64, 0i64);
v28 = *(_QWORD *)(HeaderStructure + 866);// function offset: c1b7
v29 = *(unsigned __int16 *)(HeaderStructure + 992);// function length: 4e7
return ((__int64 (__fastcall *)(_WORD *, _QWORD, _QWORD, _QWORD, _QWORD, _QWORD))RunEncryptCode)(
         v24,
         0i64,
         0i64,
         0i64,
         0i64,
         0i64);
```

Figure 8. Before executing the desired function, the process inputs its offset and length into RunEncryptCode.

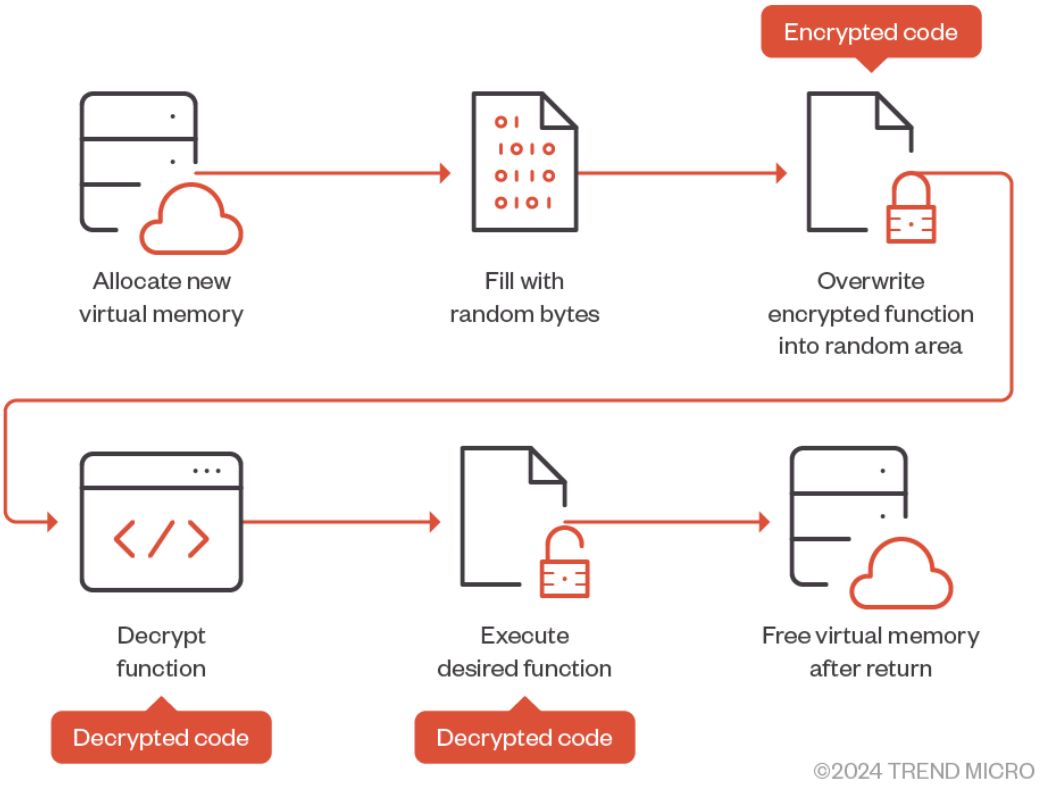
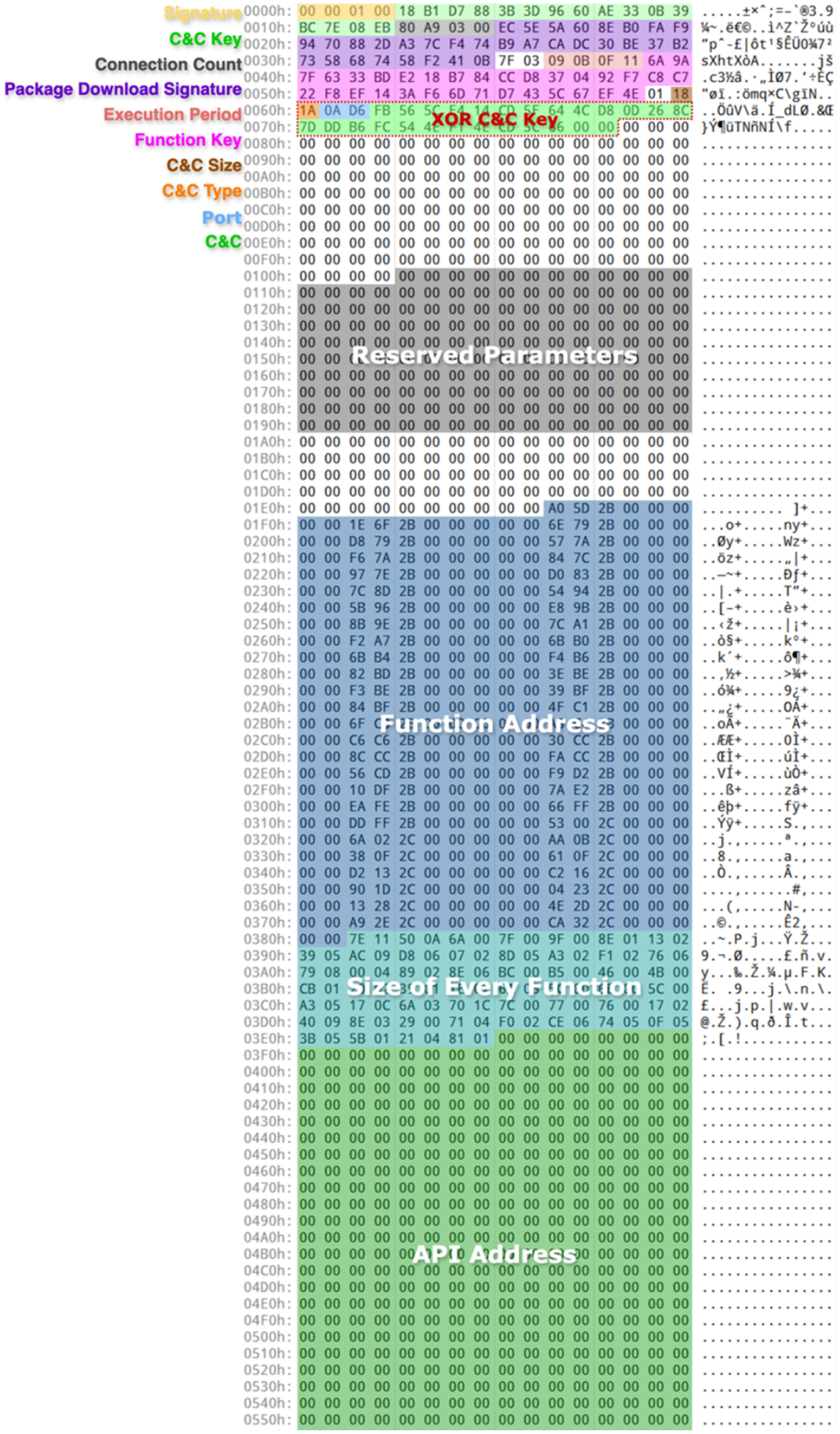


Figure 9. The flow chart of RunEncryptCode to execute desired functions

Data offset	Data size	Data content
0x00	0x04	Signature (00 00 01 00)
0x04	0x10	Key (Only for C&C decryption)
0x14	0x04	Retry connection
0x18	0x20	Signature sends to the C&C server, requesting the next-stage RAT
0x3A	0x01	Execution time lower bound in the morning (for example, 9 a.m.)
0x3B	0x01	Execution time upper bound in the morning (for example, 11 a.m.)
0x3C	0x01	Execution time lower bound in the afternoon (for example, 3 p.m.)
0x3D	0x01	Execution time upper bound in the afternoon (for example, 5 p.m.)
0x3E	0x20	Key for encrypted data and encrypted function
0x5F	0x01	(Size of encrypted C&C server) - 3
0x60	not fix	Encrypted C&C server +0: Flag for IP/Domain

		+1: Port number +3: C&C server
0x1EA	0x198	List of function address (for example, 0x8 * 51 functions)
0x382	0x66	List of function length (for example, 0x2 * 51 functions)
0x3E8	0x1A0	List of API address
0x588	0xB8	List of encrypted API hash
0x640	0x4D	List of encrypted library name

Table 8. The configuration structure of the Deuterbear downloader


Signature
C&C Key
Connection Count
Package Download Signature
Execution Period
Function Key
C&C Size
C&C Type
Port
C&C
XOR C&C Key
Reserved Parameters
Function Address
Size of Every Function
API Address

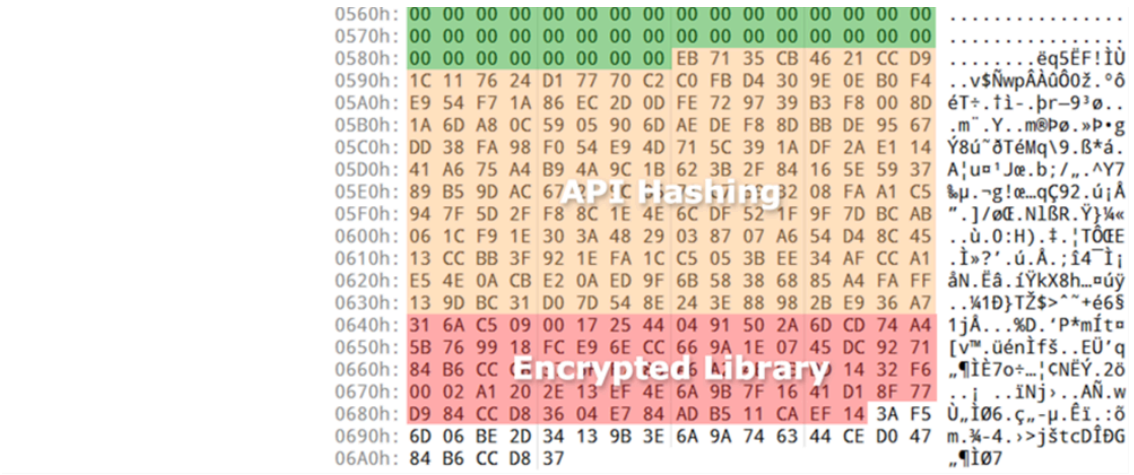


Figure 10. A screenshot showing the configuration structure of the Deuterbear downloader

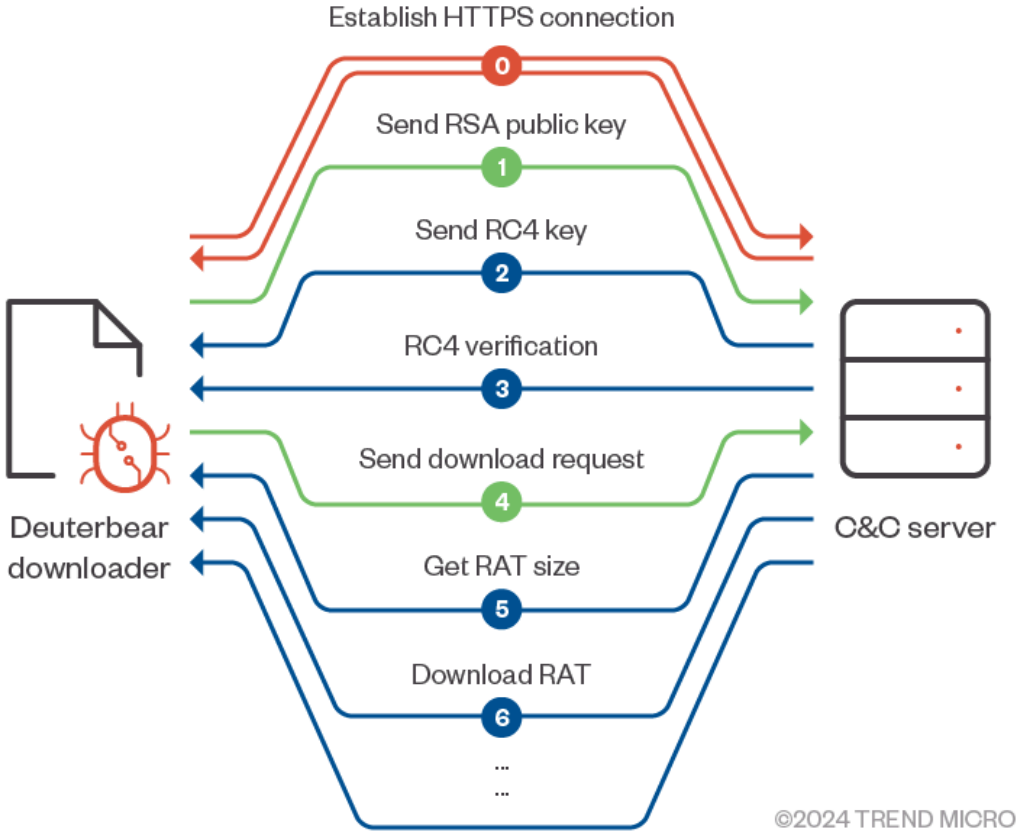


Figure 11. Network traffic to download the Deuterbear RAT

Index	Direction	Encryption	Key
1	Victim -> C&C	N/A	N/A
2	C&C -> Victim	RSA	CSP_KEY
3	C&C -> Victim	Salted RC4	RC4_KEY_2 (from index 2)
4	Victim -> C&C	Salted RC4	RC4_KEY_1 (from index 2)

5	C&C -> Victim	Salted RC4	RC4_KEY_2
6	C&C -> Victim	Salted RC4	RC4_KEY_2

Table 9. Basic information on traffic to download Deuterbear RAT

Deuterbear uses only 5 bytes in the header to describe the data information, with the general format being the following:

Offset	Size	Content
0x00	0x01	Possibly the type of packet
0x01	0x02	Command code (Like 40 1F in the packet of Waterbear downloader)
0x03	0x02	Size of data

Table 10. Header format of the Deuterbear packet

Send RSA public key

The downloader applies Microsoft CryptoAPI to generate an RSA public/private key, sending the public key to the C&C server for RSA encryption during the next communication.

The packet format is as follows:

Offset	Size	Type	Content
0x00	0x05	Header	01 CD 03 ?? ?? (The last 2 bytes are size of data with little-endian)
0x05	0x114	Data	RSA public key BLOBs for packet encryption in the next step.

Table 11. Packet format for sending the RSA public key

Send RC4 Key

The C&C server prepares two keys for RC4 encryption, RC4_KEY_1 and RC4_KEY_2. The former is applied to encrypt the traffic from the victim to the C&C server, and the latter is for the direction from the C&C server to the victim. The keys are then encrypted by RSA public generated from the victim side and sent to the victim with the following packer format:

Offset	Size	Type	Content
0x00	0x05	Header	?? CD 03 ?? ?? (The last 2 bytes are size of data with little-endian)
0x05	0x20	Data	0x05: RC4_KEY_1 0x15: RC4_KEY_2

Table 12. Packet format for sending RC4 key

RC4 verification

The victim side verifies whether the RC4 decryption is working by checking the decrypted data, which is the RSA public key.

Offset	Size	Type	Content
0x00	0x05	Header	?? ?? ?? ?? ?? (The last 2 bytes are size of data with little-endian)
0x05	0x114	Data	RSA public key BLOBs generated from victim.

Table 13. Packet format for RC4 verification

Send download request

The victim side encrypts the download signature, which is located at configuration [0x18:0x38] and sends it to the C&C server to request the next-stage shellcode.

Offset	Size	Type	Content
0x00	0x05	Header	00 CD 03 20 00 (The last 2 bytes are size of data with little-endian)
0x05	0x20	Data	The download signatures

Table 14. Packet format about sending download command to C&C

Get RAT Size

The C&C server sends the packet for the RAT size with the following format:

Offset	Size	Type	Content
0x00	0x05	Header	02 D0 03 04 00
0x05	0x04	Data	This size of RAT with little-endian

Table 15. Packet format for retrieving the RAT size

Download RAT

The C&C server sends the packet for the RAT with the following format:

Offset	Size	Type	Content
0x00	0x05	Header	01 D0 03 ?? ?? (The last 2 bytes are size of data with little-endian)

0x05	Not fixed	Data	RSA public key for packet encryption from C&C to victim

Table 16. Packet format for downloading the RAT

This step repeatedly receives the packet from the C&C server until the whole RAT is delivered. The received Deuterbear RAT is in a shellcode format, unlike the original Waterbear downloader that loads the PE file for the next-stage RAT.

Comparison

Table 17 shows the difference between the Deuterbear downloader and Waterbear downloader:

Properties	**Deuterbear downloader**	**Waterbear downloader**
Executable time	Limited	Any time
Anti-Memory scanning	Encrypt/Decrypt function in new virtual memory	Encrypt/Decrypt function in local address
Encrypted downloader path	Registry	File/Registry
Encrypted downloader decryption	CyprtUnprotectData	Salted RC4 or CyprtUnprotectData
C&C string decryption	XOR with 16-bytes key	XOR with 0xFF
C&C communication	HTTPS	HTTP
Size of packet header	5	10
Magic bytes in header	CD 03	40 1F
	D0 03	43 1F
RC4 key in downloading traffic	Generated by the C&C server	Generated by the victim
Format of downloaded RAT	Shellcode	PE file

Table 17. Differences between the Deuterbear downloader and Waterbear downloader

Conclusion

Since 2009, Earth Hundun has continuously evolved and refined the Waterbear backdoor, as well as its many variants and branches. Despite available solutions, the enhancements in infection methods and anti-analysis mechanisms have led to the most advanced variant so far — Deuterbear. The Deuterbear downloader employs HTTPS encryption for network traffic protection and implements various updates in malware execution, such as altering the function decryption, checking for debuggers or sandboxes, and modifying traffic protocols.

According to our telemetry, Earth Hundun has continued to infiltrate the Asia-Pacific region, and the ongoing evolution of Waterbear and Deuterbear presents formidable challenges to organizational defense efforts. As such, Trend Micro remains committed to further enhancing our monitoring and detection methods accordingly.

MITRE ATT&CK

Tactic	Technique	ID	Description
Execution	Shared Modules	T1129	Dynamically loads the DLLs through the shellcode
	Native API	T1106	Dynamically loads the APIs through the shellcode
Persistence	Hijack Execution Flow: DLL Side-Loading	T1574.002	Uses modified legitimate executable to load the malicious DLL
	Boot or Logon Autostart Execution: Print Processors	T1547.012	Abuses print processors to run malicious DLLs during system
Defense Evasion	Obfuscated Files or Information: Binary Padding	T1027.001	Padding huge 0x00 in encrypted downloader
	Masquerading: Match Legitimate Name or Location	T1036.005	Makes the patched executable that appears legitimate or benign to users and/or security tools
	Deobfuscate/Decode Files or Information	T1140	Uses RC4 or CryptUnprotectData to decrypt encrypted downloader
	Execution Guardrails	T1480	Targets specific path/registry in the victim's environment
	Virtualization/Sandbox Evasion: Time Based Evasion	T1497.003	Downloaders check sandbox by API, Sleep, whether normal operation.
	Debugger Evasion	T1622	Downloaders check debugger mode by process time.
Discovery	File and Directory Discovery	T1083	RAT searches files and directories or in specific locations.
	System Network Configuration Discovery: Internet Connection Discovery	T1016.001	Downloaders check for internet connectivity on compromised systems.

	System Network Connections Discovery	T1049	Waterbear RAT lists network connections to or from the compromised system they are currently accessing or from remote systems by querying for information over the network.
	Process Discovery	T1057	Waterbear RAT searches specific process.
	System Information Discovery	T1082	Waterbear RAT gets detailed information about the operating system and hardware, including version, username, and architecture.
	Query Registry	T1012	Queries data from registry to decrypt downloader
Collection	Data from Local System	T1005	Collects basic information of victim
Exfiltration	Exfiltration Over Command-and-Control Channel	T1041	Sends collected data to C&C
Command and Control	Application Layer Protocol: Web Protocols	T1071.001	Downloaders communicate with C&C by HTTP/HTTPS
	Encrypted Channel	T1573	Employs a RC4/RSA to conceal command and control traffic
	Data Encoding: Non-Standard Encoding	T1132.002	Encodes traffic with a non-standard RC4 to make the content of traffic more difficult to detect

Indicators of Compromise

The indicators of compromise for this entry can be found here.

*We'd like to thank Trend's **Dove Chiu** and **Shih-hao Weng** for additional intelligence.*

Tags
