# Noodle RAT: Reviewing the Backdoor Used by Chinese-Speaking Groups

⋮ 6/11/2024

Malware

This blog entry provides an analysis of the Noodle RAT backdoor, which is likely being used by multiple Chinese-speaking groups engaged in espionage and other types of cybercrime.

By: Hara Hiroaki June 11, 2024 Read time: 12 min (3123 words)

*This blog is based on our presentation at Botconf 2024. It can be viewed here.*

## Introduction

Since 2022, we have been investigating numerous targeted attacks in the Asia-Pacific region that used the same ELF backdoor. Most vendors identify this backdoor as a variant of existing malware such as Gh0st RAT or Rekoobe. However, we unearthed the truth: this backdoor is not merely a variant of existing malware, but is a new type altogether. We suspect it is being used by Chinese-speaking groups engaged in either espionage or cybercrime. We dubbed this formerly undocumented malware as “Noodle RAT.”

## Noodle RAT

Noodle RAT, also known as ANGRYREBEL or Nood RAT, is a relatively simple backdoor confirmed to have both Windows (Win.NOODLERAT) and Linux (Linux.NOODLERAT) versions. The following timeline details a brief history of this malware.

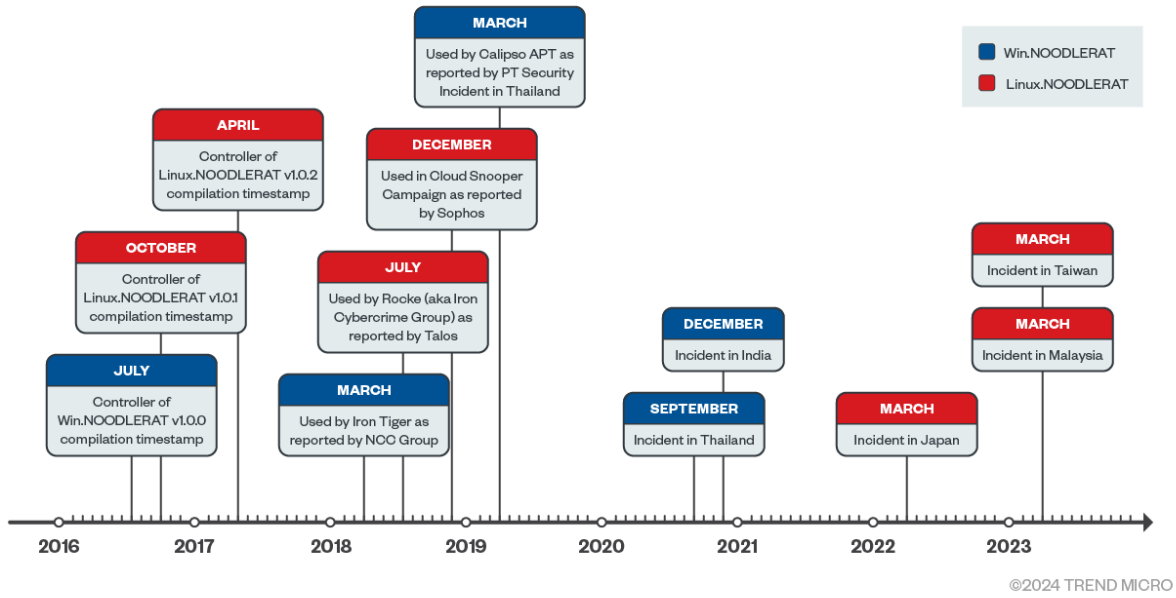



Figure 1. Noodle RAT Timeline

We uncovered in VirusTotal a couple of command-and-control (C&C) panels pointing to the early versions of Noodle RAT; v1.0.0 for Win.NOODLERAT was compiled in July 2016, v1.0.1 for Linux.NOODLERAT in December 2016, and v1.0.1 for Linux.NOODLERAT in April 2017.

Despite its long history, Noodle RAT has not been properly classified until recently. Since 2018, multiple reports have been published about attacks involving Noodle RAT, but back then, this ELF backdoor was inadvertently identified as different malware families. For instance, NCC Group released a report on a variant of Gh0st RAT used by Iron Tiger in 2018. Talos released a report on an ELF backdoor used by Rocke (aka Iron Cybercrime Group) in 2018. Sophos released a report on a Linux version of the Gh0st RAT variant used in the Cloud Snooper Campaign in 2018. Positive Technology Security released a report on Calypso RAT used by Calypso APT in 2019. Upon analysis, we discovered that the ELF backdoor mentioned in these reports was actually Noodle RAT.

Additionally, our telemetry also found espionage campaigns using Noodle RAT targeting Thailand, India, Japan, Malaysia, and Taiwan since 2020. This brief history shows that Noodle RAT has been shared among multiple groups and used for both espionage and cybercrime.

In the following sections, we discuss the details of both the Windows and Linux versions of Noodle RAT.

## Win.NOODLERAT

Win.NOODLERAT is a shellcode-formed in-memory modular backdoor, originally reported by NCC Group and Positive Technology Security. Based on other vendor's reports and our observation, it seems like Win.NOODLERAT is used by Iron Tiger, Calypso APT, and several unknown clusters in espionage campaigns. The built-in backdoor capabilities are quite simple

- Download and upload files
- Run additional in-memory modules
- Work as TCP proxy

Since Win.NOODLERAT is shellcode-formed, it requires its own loader. Our investigation has confirmed that MULTIDROP and MICROLOAD were used.

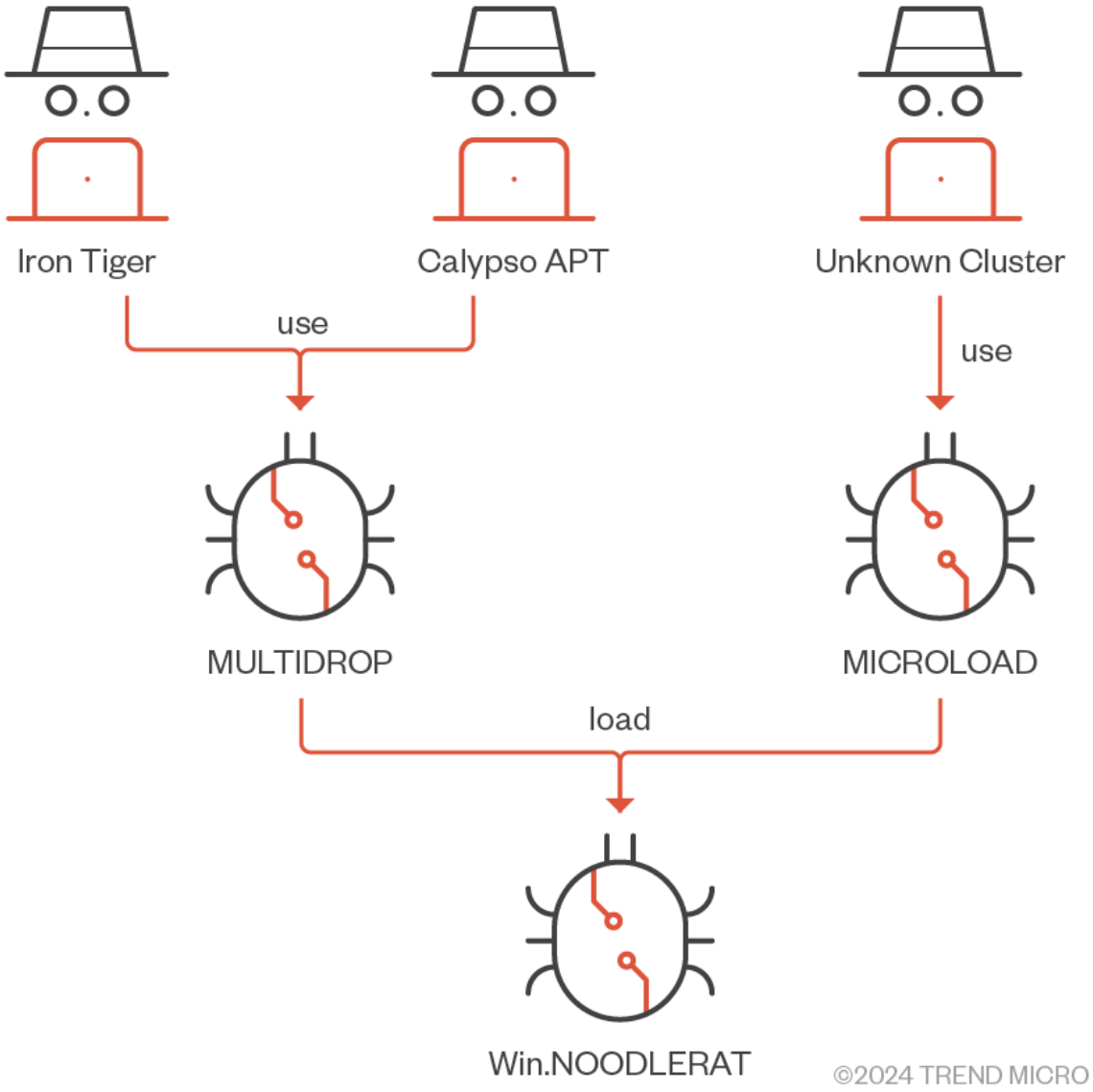



Figure 2. Relations of Win.NOODLERAT with threat groups

**Installation**

MULTIDROP, which we have observed in an espionage campaign targeting Thailand in 2019, is exactly the same as the dropper described by Positive Technology Security.

On the other hand, MICROLOAD, which we have observed in espionage campaigns targeting India from 2019 to 2021, has a different design. MICROLOAD is loaded by the legitimate Microsoft application *Oleview.exe*. It also decrypts the encrypted payload in *HKCR\Microsoft.System.UpdateColl\UpdateAgent* by RC4 and injects the decrypted shellcode into *svchost.exe*.

The first stage of the shellcode is just a decoding stub to decode the following XORed data. Once initialization is completed, the core payload and config are decoded in memory.

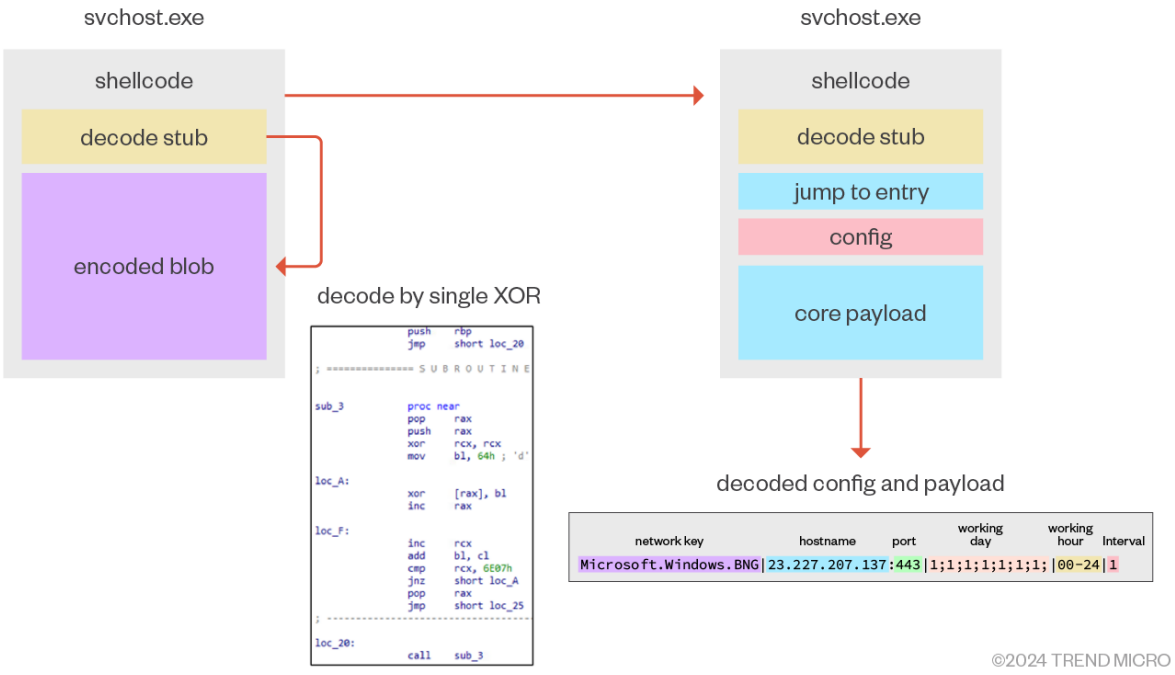



Figure 3. Initialization routine of Win.NOODLERAT

### C&C Communication

Regarding C&C communication, Win.NOODLERAT supports TCP, SSL, and HTTP. The entire communication is encrypted by RC4 and a custom algorithm using XOR and AND instructions. This algorithm, well documented by Positive Technology Security, is worth examining again as it plays a part in the comparison between Win.NOODLERAT and Linux.NOODLERAT.

The encrypted packet consists of two sections as follows: the header and the payload.

```
struct Packet {
    struct Header {
        DWORD rand1;
        DWORD rand2;
        DWORD payload_len;
        DWORD const_val;
        DWORD rand3;
        DWORD cmd_id;
        DWORD header_key;
    };
    BYTE payload[0x2000];
};
```

Figure 4. Pseudo structure of the packet

When sending data, it encrypts the header section of the packet using the following steps:

1. Initialize Header section fields *of cmd_id, payload_len, and const_val*
2. Generate 3 random DWORD values
3. Generate a part of the RC4 key by adding payload data length and random DWORD
4. Encode the value generated in Step 3 by XOR with the hardcoded key
5. Convert the values generated in Steps 3 and 4 into HEX string, and concatenate them
6. Encrypt the first 24 bytes of the header (= excluding header_key) by RC4 with the key generated in Step 5

```
pkt.field_8_cmd_id = cmd_id;
pkt.field_0_payload_len = data_len;                                  } Step #1
pkt.field_10_const = 0x7C05;
pkt.field_4_rand_1 = api->kernel32_GetTickCount() % 0x3FC7;
pkt.field_C_rand_2 = api->kernel32_GetTickCount() % 0x59AD;          } Step #2
pkt.field_14_rand_3 = api->kernel32_GetTickCount() % 0x69E0;
rand = api->kernel32_GetTickCount();
*pkt.field_18_header_key = data_len + rand;                          } Step #3
xored_rand[3] = (data_len + rand) ^ 0x5C;
xored_rand[1] = ((data_len + rand) >> 8) ^ 0xE6;
xored_rand[0] = ((data_len + rand) >> 24) ^ 0x2D;                    } Step #4
xored_rand[2] = ((data_len + rand) >> 16) ^ 0xA9;
(api->ntdll_memset)(key1, 0, 100);
format_hex_string_2(this, pkt.field_18_header_key, xored_rand, key1); Step #5
do_rc4(this, &pkt, 24, key1);                                        Step #6
```

Figure 5. Encryption algorithm of the header section

For the payload section, it encrypts as follows:

1. Copy DWORD values from the head of the header section, cmd_id filed, and cont_val field
2. Encode each DWORD in Step 1 by XOR and AND instructions
3. Convert the values generated in Step 2 into HEX string, and concatenate them
4. Encrypt the payload data by RC4 with the key generated in Step 3

```
(api->ntdll_memcpy)(key2_part3, &pkt, 4);
(api->ntdll_memcpy)(key2_part1, &pkt.field_8_cmd_id, 4);             } Step #1
(api->ntdll_memcpy)(key2_part2, &pkt.field_10_const, 4);
key2_part3[3] = key2_part3[0] ^ 0xAC;
key2_part3[0] = (key2_part3[0] ^ 0xAC) & 0xCD;
key2_part3[2] = key2_part3[1] & 0x89;
key2_part3[1] = key2_part3[1] & 0x89 ^ 0x60;
key2_part1[3] = key2_part1[0] & 0x80;
key2_part1[0] = key2_part1[0] & 0x80 ^ 0xD1;
key2_part1[2] = key2_part1[1] ^ 0x8D;                                } Step #2
key2_part1[1] = (key2_part1[1] ^ 0x8D) & 0x64;
key2_part2[3] = key2_part2[0] & 0x84;
key2_part2[0] &= 0x94u;
key2_part2[2] = key2_part2[1] ^ 0x91;
key2_part2[1] ^= 0xF9u;
(api->ntdll_memset)(key2, 0, 100);
format_hex_string_3(this, key2_part1, key2_part2, key2_part3, key2); Step #3
(api->ntdll_memcpy)(send_buf, &pkt, 28);
if ( data_len > 0 )
  do_rc4(this, payload, data_len, key2);                             Step #4
```

Figure 6. Encryption algorithm of the payload section

**Backdoor Commands**

During our analysis, we discovered that there are different types of Win.NOODLERAT that implement various command IDs. Based on one of the command IDs received upon successful authentication by the C&C server, we categorized them into two clusters: Type 0x03A2 and Type 0x132A. The backdoor capability is implemented using a combination of major-ID and optional sub-ID. Table 1 lists the backdoor commands:

| Actions | Type 0x03A2 | | Type 0x132A | |
|---|---|---|---|---|
| | Major-ID | Sub-ID | Major-ID | Sub-ID |
| Successfully authorized | 0x03A2 | - | 0x132A | - |
| Message of the end of command | 0x0AC3 | - | 0x1AC3 | - |
| Initialize module metadata | 0x194C | - | 0x294C | - |
| Receive module data | 0x1AF2 | - | 0x2AC8 | - |
| Launch module without pipe | 0x1397 | - | 0x230E | - |
| Delete module metadata | 0x1D50 | - | 0x2D06 | - |
| Upload a file to C&C server | 0x390A | 0x35C3 & 0x35C4 & 0x3013 | 0x590A | 0x55C3 & 0x55C4 & 0x5013 |
| List directories recursively | 0x390A | 0x35C5 | 0x590A | 0x55C5 |
| Download a file from C&C server | 0x390A | 0x35C7 & 0x35C8 & 0x35C9 & 0x3013 | 0x590A | 0x55C7 & 0x55C8 & 0x55C9 & 0x5013 |
| Write given data to pipe | 0x2099 | 0x2186 | 0x3099 | 0x3167 |
| Write 0x32E0 to pipe | 0x2099 | 0x220E | 0x3099 | 0x32E0 |
| Write 0x38AF to pipe | 0x2099 | 0x28FA | 0x3099 | 0x38AF |
| Send module data to another module | 0x2099 | 0x2741 | 0x3099 | 0x3716 |
| Same as 0x3099 | 0x2099 | 0x2A0B | 0x3099 | 0x3A0B |
| Start TCP server to proxy packets to the C&C server | 0x2099 | 0x2CBD | 0x3099 | 0x3CD0 |
| Delete itself | N/A | | 0x1C1C | - |

Table 1. Backdoor commands of Win.NOODLERAT

The first one, Type 0x03A2, implements most commands except the last one, deleting itself. This type of Win.NOODLERAT was used by Iron Tiger and other unknown clusters for espionage purposes,

suggesting that this version could be a shared version.

The second one, Type 0x132A, implements full features. This type of Win.NOODLERAT was used only by Calypso APT. Therefore, this is likely an exclusive version.

Interestingly, upon comparing the command IDs, we found that some have similar parts. For instance, the command IDs to upload a file to the C&C server are 0x390A and 0x590A respectively; this similarity might be an indicator of versioning, but there is not enough evidence to conclude such.

**Linux.NOODLERAT**

Linux.NOODLERAT is an ELF version of Noodle RAT, but with a different design. This backdoor has been used by several groups with various motivations, such as Rocke (Iron Cybercrime Group) for financial gains

Cloud Snooper Campaign for espionage, and an unknown cluster also for spying purposes. Since it's designed differently, its backdoor capabilities are also slightly different:

- Reverse shell
- Download & Upload files
- Scheduling execution
- SOCKS tunneling

**Initialization**

In most cases, Linux.NOODLERAT was deployed as an additional payload of an exploit against public-facing applications. After deployment, the backdoor copies itself to /tmp/CCCCCCCC and performs process name spoofing by overwriting "argv." Then, it decrypts the embedded config by RC4 with the hardcoded key, "r0st@#$." The decrypted config is formatted as shown in the diagram below; Linux.NOODLERAT will connect to the defined C&C server based on the config.

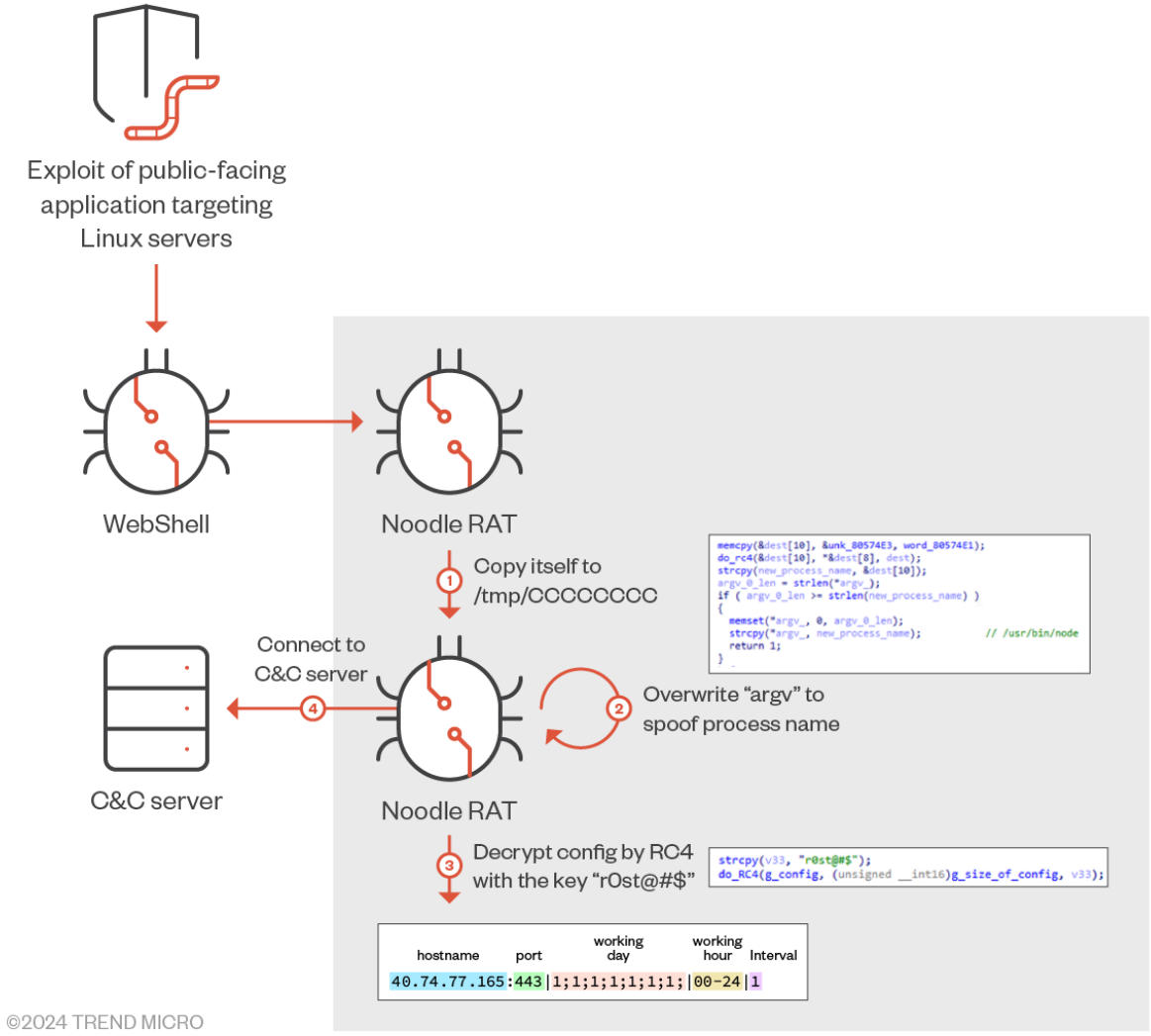



Figure 7. Observed execution flow of Linux.NOODLERAT

### C&C Communication

Linux.NOODLERAT supports only TCP and HTTP for C&C communication protocol. Interestingly, the backdoor uses two different algorithms to encrypt the communication: the first part is for command processing, which uses RC4 and XOR+AND; and the second part is for a reverse shell session, which uses HMAC_SHA1 and AES128-CBC.

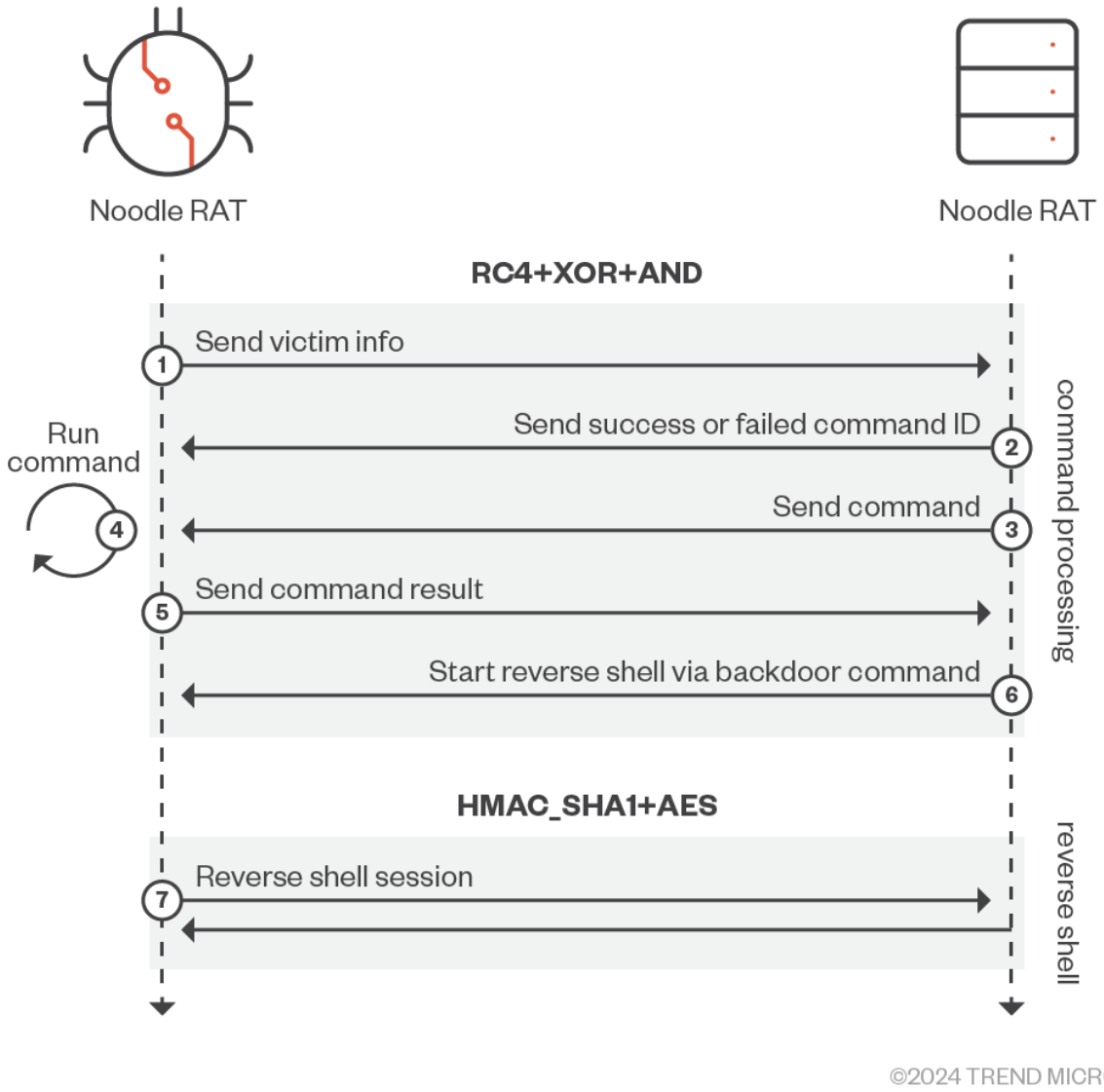



Figure 8. Overview of C&C communication of Linux.NOODLERAT

The encryption algorithm in the command processing session uses RC4 and a combination of XOR + AND instructions. This is mostly the same with Win.NOODLERAT even though the hardcoded key and APIs are different, as the following images show:

Linux.NOODLERAT

```
*pkt.field_14_cmd_id = cmd_id;
*pkt.field_8_payload_len = data_len | 0x9F7200000000LL;
pkt.field_4_rand_2 = gen_random() % 0x5C39;
pkt.field_10_rand_3 = gen_random() % 0xF71A;
pkt.field_0_rand_1 = gen_random() % 0x80B2;
rand = gen_random();
*pkt.field_18_header_key = data_len + rand;
xored_rand[3] = ((data_len + rand) ^ 0xB5) + 7;
xored_rand[2] = (((data_len + rand) >> 16) ^ 0x79) + 5;
xored_rand[1] = (((data_len + rand) >> 8) ^ 0x3D) + 1;
xored_rand[0] = (((data_len + rand) >> 24) ^ 0x9A) + 3;
memset(key1, 0, sizeof(key1));
v24 = 0;
format_hex_string_2(pkt.field_18_header_key, xored_rand, key1);
do_rc4(&pkt, 24, key1);
```

Win.NOODLERAT

```
pkt.field_8_cmd_id = cmd_id;
pkt.field_0_payload_len = data_len;
pkt.field_10_const = 0x7C05;
pkt.field_4_rand_1 = api->kernel32_GetTickCount() % 0x3FC7;
pkt.field_C_rand_2 = api->kernel32_GetTickCount() % 0x59AD;
pkt.field_14_rand_3 = api->kernel32_GetTickCount() % 0x69E0;
rand = api->kernel32_GetTickCount();
*pkt.field_18_header_key = data_len + rand;
xored_rand[3] = (data_len + rand) ^ 0x5C;
xored_rand[1] = ((data_len + rand) >> 8) ^ 0xE6;
xored_rand[0] = ((data_len + rand) >> 24) ^ 0x2D;
xored_rand[2] = ((data_len + rand) >> 16) ^ 0xA9;
(api->ntdll_memset)(key1, 0, 100);
format_hex_string_2(this, pkt.field_18_header_key, xored_rand, key1);
do_rc4(this, &pkt, 24, key1);
```

Figure 9. Encryption algorithm of the header section

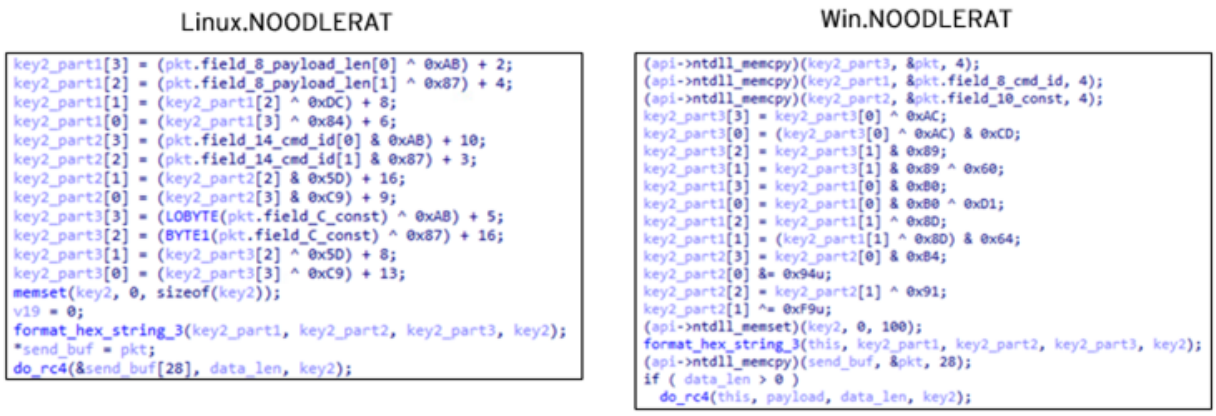



Figure 10. Encryption algorithm of the payload section

The reverse shell session will be initiated when it receives a backdoor command. Since the encryption algorithm in this session, as we will describe later, is totally the same as in one of the existing malware, we will explain the authentication mechanism as simply as possible:

1. It receives 40 bytes of data from the C&C server, which will be divided into two chunks of 20 bytes. Each chunk is then combined with the hardcoded string and the SHA1 hash value is calculated; HMAC_SHA1 pairs will be used for the AES key and IV of encryption/decryption and integrity check.
2. It receives an additional 16 bytes of data from the C&C server and decrypts it with the generated AES key materials.
3. It verifies if the decrypted challenge is the same as the hardcoded challenge bytes to ensure the C&C server's authenticity.

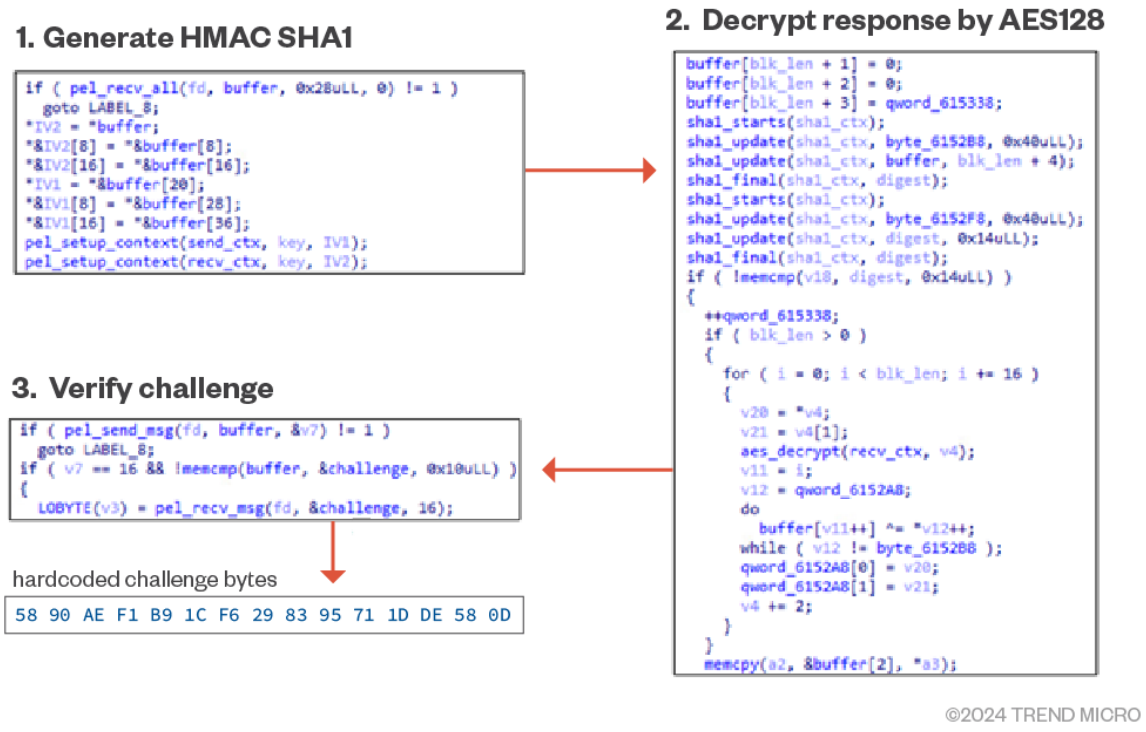



Figure 11. Authentication mechanism of Linux.NOODLERAT

**Backdoor Commands**

Like Win.NOODLERAT, we found that there are also two clusters of Linux.NOODLERAT based on the implemented backdoor command IDs: Type 0x03A2 and Type 0x23F8.

| **Actions** | **Type 0x03A2** | | **Type 0x23F8** | |
|---|---|---|---|---|
| | **Major-ID** | **Sub-ID + arg (optional)** | **Major-ID** | **Sub-ID + arg (optional)** |
| Successfully authorized | 0x03A2 | - | 0x23F8 | - |
| Start backdoor routine | 0x2099 | - | 0x4799 | - |
| Initiate reverse shell session | 0x1 | 0x2186 | 0x1 | 0x5186 |
| Show lists of files and directories in root dir | 0x2 | 0x2186 + 0x1 | 0x2 | 0x5186 + 0x1 |
| Show lists of files and directories in specific dir | 0x2 | 0x2186 + 0x2 | 0x2 | 0x5186 + 0x2 |
| Rename a file | 0x2 | 0x2186 + 0x3 | 0x2 | 0x5186 + 0x3 |
| Remove a file | 0x2 | 0x2186 + 0x4 | 0x2 | 0x5186 + 0x4 |
| Upload a file to the C&C server | 0x3 | 0x35C3 & 0x35C4 & 0x3013 | 0x3 | 0x35C3 & 0x35C4 & 0x3013 |
| List directories recursively | 0x3 | 0x35C5 | 0x3 | 0x35C5 |
| Download a file from the C&C server | 0x3 | 0x35C7 & 0x35CB & 0x35CC & 0x3013 | 0x3 | 0x35C7 & 0x35CB & 0x35CC & 0x3013 |
| Create a new directory | 0x3 | 0x35C8 & 0x35C9 & 0x35CA & 0x35CD | 0x3 | 0x35C8 & 0x35C9 & 0x35CA & 0x35CD |
| Initiate SOCKS tunneling | 0x4 | - | 0x4 | - |
| Initiate SOCKS tunneling | 0x5 | - | 0x5 | - |
| Show current datetime | 0x6 | 0x2186 + 0x1 | N/A | 0x5186 + 0x1 |
| Download a file as /usr/include/sdfwex.h or /tmp/.llock | 0x6 | 0x2186 + 0x2 | N/A | 0x5186 + 0x2 |

Table 2. Backdoor commands of Linux.NOODLERAT

For those marked with N/A, functions for these commands are implemented but there is no way to jump since Major-ID for these commands is not supported.

Compared to Win.NOODLERAT, most of the backdoor commands are different except the command for the response of successful authorization in Type 0x03A2. Also, Linux.NOODLERAT requires arguments for some specific commands. Like Win.NOODLERAT, comparing the commands between Type 0x03A2 and Type 0x23F8, both variants have a similar command sub-ID, such as 0x2186 in Type 0x03A2 and 0x5186 in Type 0x23F8. As previously stated, there is a chance that this might be an indicator of versioning. For the attribution, Type 0x03A2 was used by Rocke, Cloud Snooper campaign, and other unknown clusters, which possibly means that this type might be a shared version. On the other hand, more context is needed to fully attribute Type 0x23F8 to a specific group.

**Similarities between Win.NOODLERAT and Linux.NOODLERAT**

We have already mentioned that Linux.NOODLERAT shares mostly the same code in the C&C communication process with Win.NOODLERAT. Adding to that, both variants have several other similarities/overlaps. First is the backdoor command. As stated earlier, both Win.NOODLERAT and Linux.NOODLERAT expect to receive the command “0x3A2” when it’s authorized by the C&C server.

Linux.NOODLERAT

```
memset(&victim_info, 0, 0x258uLL);
get_victim_info(&victim_info);
if ( (int)send_data(&victim_info, 520, 0x2E9, a2, 0) <= 0
  || (memset(&readfds, 0, sizeof(readfds)),
      field_270_main_socket = a2->field_270_main_socket,
      readfds.fds_bits[field_270_main_socket / 64] |= 1LL << (field_270_main_socket % 64),
      timeout.tv_sec = 20LL,
      timeout.tv_usec = 0LL,
      select(field_270_main_socket + 1, &readfds, 0LL, 0LL, &timeout) <= 0)
  || (memset(v7, 0, sizeof(v7)), (int)recv_data(v7, 10240, &cmd_id, a2, 0) <= 0)
  || (result = 1LL, cmd_id != 0x3A2) )
{
  close_sock(&a2->field_270_main_socket);
  do_sleep(60000 * a2->field_297_interval);
  v5 = *a1 + 1;
  *a1 = v5;
  field_268_max_count = a2->field_268_max_count;
  result = 0LL;
  if ( v5 >= field_268_max_count )
  {
    *a1 = v5 % field_268_max_count;
    return 0LL;
  }
}
return result;
```

Win.NOODLERAT

```
api = this->api;
(this->api->ntdll_memset)(this, &victim_info, 0, 400);
get_victim_info(this, &victim_info);
if ( send_data(214, this, &victim_info, 0x2E9, 0) > 0 )
{
  v7.fd_array[0] = this->config.main_socket;
  v7.fd_count = 1;
  v8.tv_sec = 20;
  v8.tv_usec = 0;
  if ( api->ws2_32_select(0, &v7, 0, 0, &v8) > 0 )
  {
    (api->ntdll_memset)(v5, 0, 10240);
    if ( recv_data(this, v5, 10240, &cmd_id, 0) > 0 && cmd_id == 0x3A2 )
      return 1;
  }
}
close_socket(this, &this->config.main_socket);
random_sleep(60000 * this->config.field_398_interval, this);
++*v10;
field_369_max_count = this->config.field_369_max_count;
if ( *v10 >= field_369_max_count )
  *v10 %= field_369_max_count;
return 0;
```

Figure 12. Same backdoor command for Linux.NOODLERAT and Win.NOODLERAT

Another overlap is the config format. Win.NOODLERAT and Linux.NOODLERAT use mostly the same format for config data except the network key section in Win.NOODLERAT.

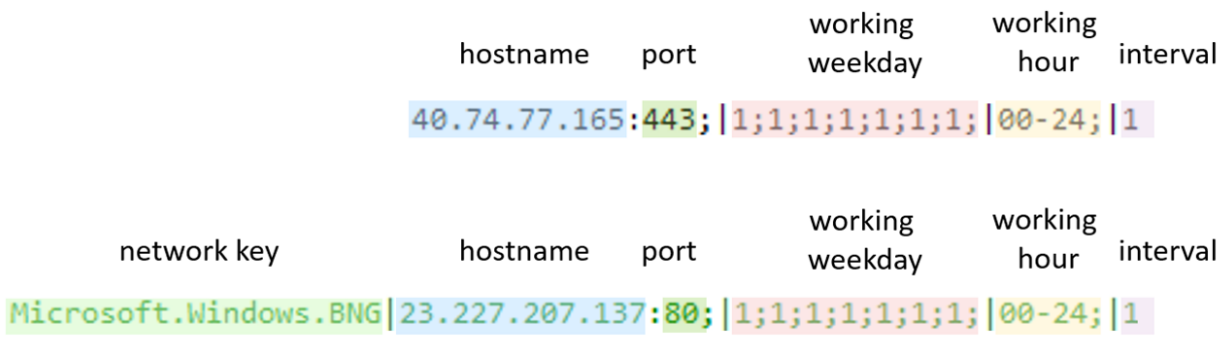



Figure 13. Win.NOODLERAT and Linux.NOODLERAT using similar config formats

# New Genes? Or Just a Legacy?

Up to this point, we have analyzed two already-documented backdoors that support different operating systems, suggesting that they belong to the same new malware family, Noodle RAT. However, several vendors have mentioned that Noodle RAT could be a variant of existing malware such as Gh0st RAT or Rekoobe. Their conclusion raises a question: Can Noodle RAT really be considered as new malware? To answer this, let's revisit Gh0st RAT and Rekoobe.

Gh0st RAT was originally developed by the C. Rufus Security Team in China. Due to a leakage in 2008, the source code of Gh0st RAT has been made publicly available. Therefore, nowadays there are so many variants of Gh0st RAT being used by multiple threat actors.

Noodle RAT has several links to Gh0st RAT. For instance, as NCC reported in 2018, Win.NOODLERAT was using the same plugins as those used by Gh0st RAT. Additionally, Noodle RAT implemented a slightly similar packet encryption algorithm used by some variants of Gh0st RAT, such as Gh0stCringe, HiddenGh0st, and Gh0stTimes. Gh0st RAT variants often use a custom algorithm using a combination of XOR and other instructions, as Noodle RAT does.

However, apart from the plugins, there is no apparent similarity in the rest of the code of Noodle RAT and Gh0st RAT, leading us to the conclusion that the plugins were simply reused but the backdoor itself is totally different.

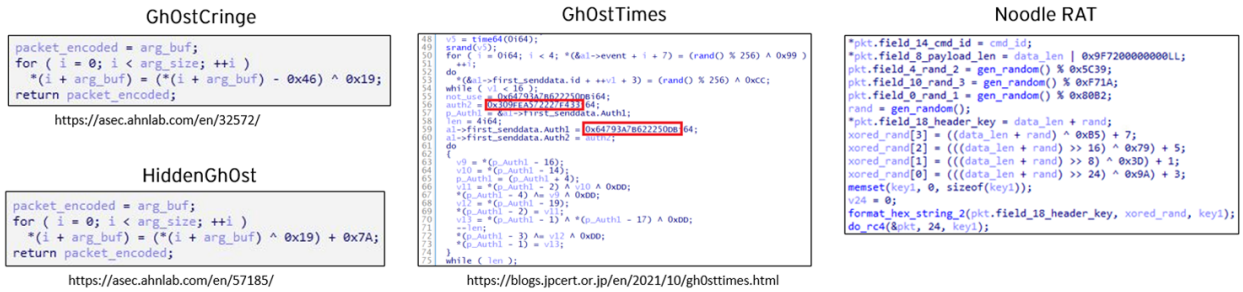



Figure 14. Algorithm comparison between Gh0st RAT variants and Noodle RAT

On the other hand, Rekoobe is a backdoor based on Tiny SHell (aka tsh) and its source code is publicly available on GitHub. It has been used for years in campaigns targeting Linux/Unix systems. This backdoor has already been detailed in reports published by several vendors. Dr.Web published the first report in 2015, and Intezer reported another version of Rekoobe in 2018. To avoid confusion, we will refer to the one mentioned by Dr.Web as "v2015" and the one mentioned by Intezer as "v2018".

Rekoobe has been used by several actors with continued updates. As Dr.Web's report mentioned, Rekoobe v2015 was used by the threat group APT31. This version modified the C&C protocol by adding a MAGIC value and changed its strategy to load the config data from the external file, not from the argument. Rekoobe v2018, based on our observation, has been use by multiple groups. This version reverted the C&C protocol into the original one but changed its strategy again to embed the config data in the binary itself. And it added a new feature to spoof the process name by overwriting "argv."

Comparing Rekoobe and Noodle RAT, one might find that some parts of Linux.NOODLERAT's code is the same as Rekoobe v2018, specifically the part of reverse shell and process name spoofing, as the

following images show:

Linux.NOODLERAT

```
if ( recv_all(fd, g_recv_buf, 0x28, 0) != 1 )
  goto LABEL_8;
*iv_2 = *g_recv_buf;
*&iv_2[4] = *&g_recv_buf[4];
*&iv_2[8] = *&g_recv_buf[8];
*&iv_2[12] = *&g_recv_buf[12];
*&iv_2[16] = *&g_recv_buf[16];
*iv_1 = *&g_recv_buf[20];
*&iv_1[4] = *&g_recv_buf[24];
*&iv_1[8] = *&g_recv_buf[28];
*&iv_1[12] = *&g_recv_buf[32];
*&iv_1[16] = *&g_recv_buf[36];
aes_hmac_sha1(&g_hmac_1, secret, iv_1);
aes_hmac_sha1(&g_hmac_2, secret, iv_2);
if ( revc_message(fd, g_recv_buf, v6) != 1 )
  goto LABEL_8;
if ( v6[0] == 16 && !memcmp(g_recv_buf, &g_challenge, 0x10u) )
{
  LOBYTE(v2) = send_message(fd, &g_challenge, 0x10u);
```

Rekoobe v2018 (Tiny SHell)

```
if ( recv_all(a1, g_recv_buf, 40u, 0) == 1 )
{
  *iv_2 = *g_recv_buf;
  *&iv_2[4] = *&g_recv_buf[4];
  *&iv_2[8] = *&g_recv_buf[8];
  *&iv_2[12] = *&g_recv_buf[12];
  *&iv_2[16] = *&g_recv_buf[16];
  *iv_1 = *&g_recv_buf[20];
  *&iv_1[4] = *&g_recv_buf[24];
  *&iv_1[8] = *&g_recv_buf[28];
  *&iv_1[12] = *&g_recv_buf[32];
  *&iv_1[16] = *&g_recv_buf[36];
  aes_hmac_sha1(g_hmac_1, secret, iv_1);
  aes_hmac_sha1(g_hmac_2, secret, iv_2);
  if ( recv_message(a1, g_recv_buf, &v6) == 1 )
  {
    if ( v6 == 16 && (v3 = memcmp(g_recv_buf, &g_challenge, 16)) == 0 )
    {
      if ( send_message(a1, &g_challenge, 16) == 1 )
```

Figure 15. Handshake on the reverse shell of Linux.NOODLERAT (left) and Rekoobe/Tiny SHell (right)

Linux.NOODLERAT

```
memcpy(&dest[10], &unk_80574E3, word_80574E1);
do_rc4(&dest[10], *&dest[8], dest);
strcpy(new_process_name, &dest[10]);
argv_0_len = strlen(*argv_);
if ( argv_0_len >= strlen(new_process_name) )
{
  memset(*argv_, 0, argv_0_len);
  strcpy(*argv_, new_process_name);          // /usr/bin/node
  return 1;
}
```

Rekoobe v2018

```
memset(*argv, 0, strlen(*argv));
strcpy(*argv, "/bin/bash");
```

Figure 16. Spoofing process name of Linux.NOODLERAT (left) and Rekoobe v2018 (right)

The code of the reverse shell session of Linux.NOODLERAT is completely the same as the one of Tiny SHell, which leads us to believe that the author of Noodle RAT might have just copied this part from Tiny SHell in GitHub. On the other hand, the technique to spoof the process name by overwriting “argv” is unique to Rekoobe v2018; this is not implemented in Tiny SHell and Rekoobe v2015. This can indicate that the author of Noodle RAT might be able to access the source code of Rekoobe v2018.

Still, since the rest of the code of Linux.NOODLERAT is totally different from any version of Rekoobe or Tiny SHell, we can conclude that Linux.NOODLERAT should be classified as another malware family.

# Server Side of Noodle RAT

In the report of NCC Group in 2019, they disclosed the control panels of Win.NOODLERAT. One of them is named “Noodlesv1.0.0,” which might indicate that it’s for Win.NOODLERAT for v1.0.0. We also recently found a new version of the control panel and builder of Linux.NOODLERAT in VirusTotal.

### Control Panel

The control panel we found was named “NoodLinux v1.0.1.” Like the previously found control panel, it requires a password to open. The password for the previous one is the current year and month (e.g., “202405”), but for this version, the password was hardcoded in the control panel. It could be opened with the password “hello!@#”.

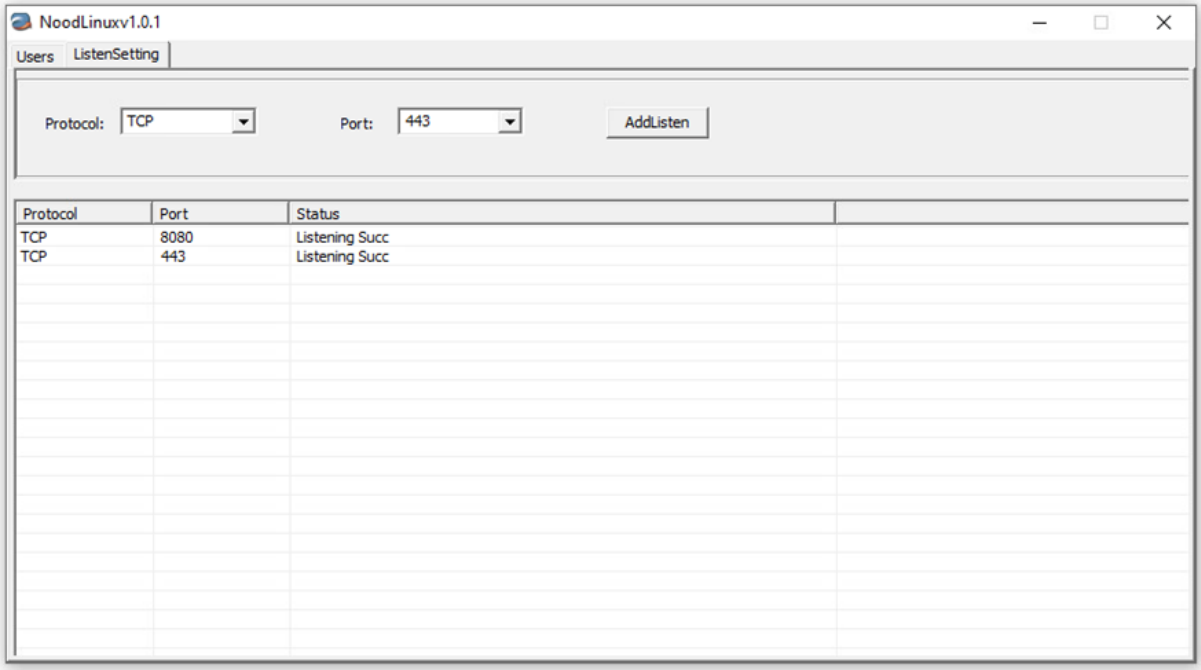



Figure 17. Control panel of Linux.NOODLERAT v1.0.1

This version of the control panel supports TCP and HTTP for C&C protocol and listens on the specified port. We tested it with the Linux.NOODLERAT sample and confirmed that it successfully works.

**Builder**

We found two versions of the builder for Linux.NOODLERAT: v1.0.1 and v1.0.2. This builder helps build Linux.NOODLERAT with a custom config defined by the user.

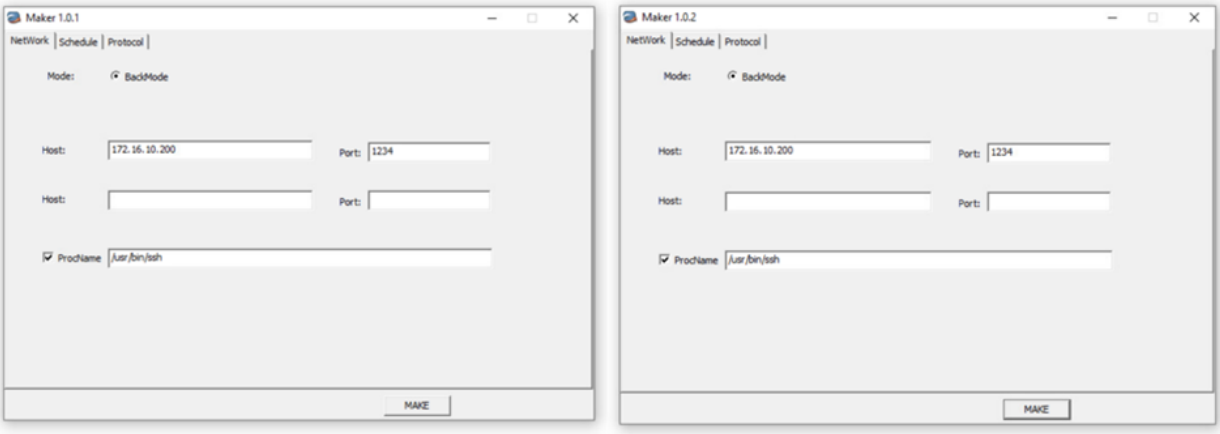
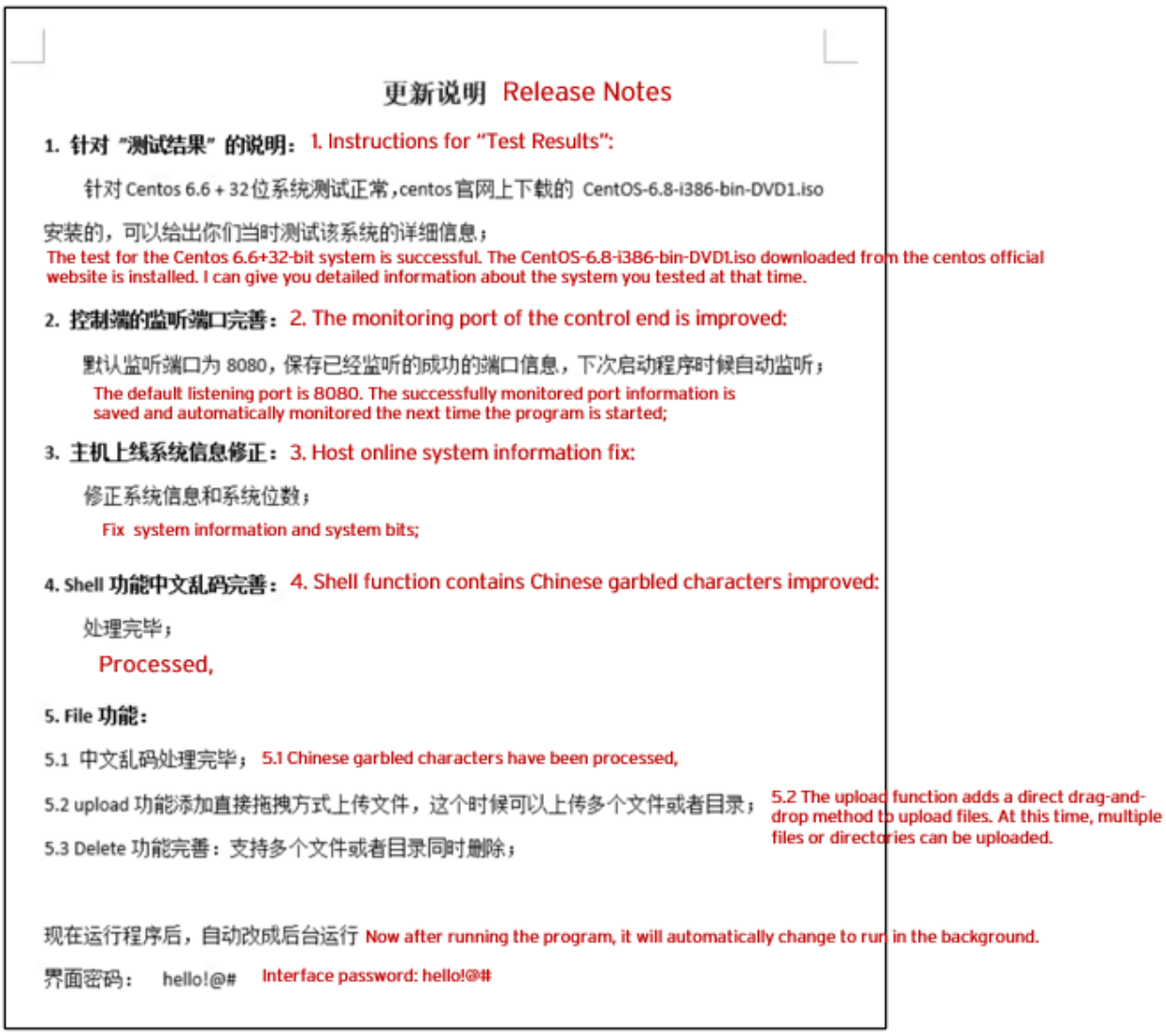



Figure 18. Builder of Linux.NOODLERAT v1.0.1 and v1.0.2

Interestingly, these server-side components were submitted with a release note in .doc form. Its file name, written in Simplified Chinese, is 更新说明, which translates to release notes or update instructions. Additionally, the name of the ini file, 副本, translates to “copy.”


Figure 19. Uploaded package containing a release document

The release note is also written in Simplified Chinese and describes the test results, improvements, fixes, and updates. This might indicate that there is a developer and a client behind this malware, like in legitimate software businesses.


更新说明 Release Notes

1. 针对 "测试结果" 的说明： 1. Instructions for "Test Results":

针对 Centos 6.6 + 32位系统测试正常，centos 官网上下载的 CentOS-6.8-i386-bin-DVD1.iso

安装的，可以给出你们当时测试该系统的详细信息；

The test for the Centos 6.6+32-bit system is successful. The CentOS-6.8-i386-bin-DVD1.iso downloaded from the centos official website is installed. I can give you detailed information about the system you tested at that time.

2. 控制端的监听端口完善： 2. The monitoring port of the control end is improved:

默认监听端口为 8080，保存已经监听的成功的端口信息，下次启动程序时候自动监听；

The default listening port is 8080. The successfully monitored port information is saved and automatically monitored the next time the program is started;

3. 主机上线系统信息修正： 3. Host online system information fix:

修正系统信息和系统位数；

Fix system information and system bits;

4. Shell 功能中文乱码完善： 4. Shell function contains Chinese garbled characters improved:

处理完毕；

Processed,

5. File 功能：

5.1 中文乱码处理完毕； 5.1 Chinese garbled characters have been processed,

5.2 upload 功能添加直接拖拽方式上传文件，这个时候可以上传多个文件或者目录； 5.2 The upload function adds a direct drag-and-drop method to upload files. At this time, multiple files or directories can be uploaded.

5.3 Delete 功能完善：支持多个文件或者目录同时删除；

现在运行程序后，自动改成后台运行 Now after running the program, it will automatically change to run in the background.

界面密码： hello!@# Interface password: hello!@#

Figure 20. Release document with English translations

# Conclusion

We have provided an in-depth analysis of Noodle RAT for both Windows and Linux, highlighting similarities and overlaps with known malware types such as Gh0st RAT and Rekoobe, and suggesting that Noodle RAT is likely shared (or for sale) among Chinese-speaking groups. Additionally, we have also revealed a control panel and builder of Noodle RAT, suggesting the possibility of an existing malware ecosystem.

Noodle RAT has been misclassified and underrated for years. We hope that this blog will help people to properly evaluate an incident related to Noodle RAT.

We have confirmed that some samples of Noodle RAT were uploaded in Virus Total in 2024, which means that it is highly probable that the malware is still in use. Considering the increase of exploitation against public-facing applications in recent years, malware targeting Linux/Unix systems is becoming more essential for attackers. It might suggest that Noodle RAT could continue to be an attractive option for threat actors for attacks.

**Trend Vision One Hunting Query**

The following query lists potentially useful queries for threat hunting within Vision One:

- Noodle RAT detection
  - malName:*NOODLERAT* AND eventName: MALWARE_DETECTION
- Potential Noodle RAT infection
  - FileFullPath:"/usr/include/sdfwex.h" OR FileFullPath:"/tmp/.llock"

Indicators of Compromise

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