Linux.Rekoobe.1

vms.drweb.com/virus/

SHA1

- a11bda0acdb98972b3dec706d35f7fba59587f99 (SPARC)
- 04f691e12af2818015a8ef68c6e80472ae404fec (SPARC)
- 466d045c3db7c48b78c6bb95873b817161a96370 (SPARC)
- cd274e6b73042856e9eec98d258a96cfbe637f6f (Intel x86)
- 8e93cfbaaf7538f8965080d192df712988ccfc54 (Intel x86-64)

A Trojan for Linux intended to infect machines with the SPARC architecture and Intel x86, x86-64 computers. The Trojan's configuration data is stored in a file encrypted with XOR algorithm. The directory of the file may be the following:

```
/usr/lib/liboop-trl.so.0.0.0
/usr/lib/libhistory.so.5.7
/usr/lib/libsagented.so.1
/usr/lib/libXcurl
/usr/lib/llib-llgrpc
```

The file has the following structure:

```
SECRET value
MAGIC value
PROXYHOST value
PROXYPORT value
USERNAME value
PASSWORD value
ENDPOINT value
SERVER_PORT value
CONNECT_BACK_DELAY value
```

Instead of the “value” variable, it contains the value of the corresponding parameter. Once data from the configuration file is received successfully, the Trojan refers to the C&C server for commands with an interval specified by the CONNECT_BACK_DELAY parameter. The address of the server is specified by the ENDPOINT parameter. If a value of the PROXYHOST parameter is not “none”, connection to the server is established via a proxy server, authorization data for which is also extracted from the configuration file.

The connection to the C&C server begins with sending of the MAGIC parameter from the configuration file and reception of a 40-byte response. Then 40 bytes are split into two blocks, which are used for AES context initialization: one block is for the received data, and the other, for the sent one:

```
int __cdecl AES_Init(st_aes_ctx *aes_ctx, char *data, char *salt)
{
  ...
  if ( RecvPacket(fd, buffer, 40, 0) != 1 )
    goto err_occured;
  *(_DWORD *)dec_salt = *(_DWORD *)buffer;
  *(_DWORD *)&dec_salt[4] = *(_DWORD *)&buffer[4];
  *(_DWORD *)&dec_salt[8] = *(_DWORD *)&buffer[8];
  *(_DWORD *)&dec_salt[12] = *(_DWORD *)&buffer[12];
  *(_DWORD *)&dec_salt[16] = *(_DWORD *)&buffer[16];
  *(_DWORD *)enc_salt = *(_DWORD *)&buffer[20];
  *(_DWORD *)&enc_salt[4] = *(_DWORD *)&buffer[24];
  *(_DWORD *)&enc_salt[8] = *(_DWORD *)&buffer[28];
  *(_DWORD *)&enc_salt[12] = *(_DWORD *)&buffer[32];
  *(_DWORD *)&enc_salt[16] = *(_DWORD *)&buffer[36];
  AES_Init(&aes_ctx_encrypt, secret, enc_salt);
  AES_Init(&aes_ctx_decrypt, secret, dec_salt);
  ...
}
```

where the AES_Init function generates an encryption key based on the SHA1 value from the “secret” parameter and the sent enc_salt or dec_salt block:

```
int __cdecl AES_Init(st_aes_ctx *aes_ctx, char *data, char *salt)
{
  ...
  sha1_init(&ctx);
  sha1_update(&ctx, data, strlen(data));
  sha1_update(&ctx, salt, 0x14u);
  sha1_final(&ctx, hash);
  AES_InitKey(aes_ctx, hash, 128);
  ...
}
```

The AES_Init function for every AES context also creates two special 40-byte blocks which are later used as a signature. For that, 40 bytes with 0x36 value are added to the verify_1 array, and 40 bytes with 0x5C value are added to the verify_2 array. Then the first 20 bytes of every array are encrypted with XOR algorithm containing the corresponding 20 bytes of the AES key.

All the later information transmitted either side during the established connection will be sent as specifically formed packages.

The first received package contains 16-byte identifier. The Trojan compares it with an identifier already stored in its body. If the match is found, the malware sends verification to the server.

Once the connection to the C&C server is established, the Trojan attempts to get a command from the server. Upon receiving a command number, the first two bytes are ignored, and the third one stands for a command identifier.

During the reception of a package from the server, the malware acquires 16 bytes, which are encrypted in AES-CBC-128 mode. The first WORD (MSB) of the received buffer is the size of the next data block (the size parameter).

After this, the Trojan calculates the package size by the “packetsize = size + 2 bytes + alignment” formula and receives the data of packetsize + 4 bytes size in the same buffer using offset of 0x10 bytes from its beginning. The last 20 bytes are the signature.

In order to verify the signature, the modification of its first DWORD is as follows: the first three bytes are replaced with zeros, and the forth one contains the package number (the Trojan records the amount of the received and sent packages in the corresponding AES contexts). Then the buffer that received the data and where DWORD was modified is used for generating of SHA1 hash (buffer is specified as “buffer”):

```
...
sha1_init(&sha1_ctx);
sha1_update(&sha1_ctx, aes_ctx_decrypt.verify_1, 0x40u);
sha1_update(&sha1_ctx, buffer, size + 4);
sha1_final(&sha1_ctx, &hash);
sha1_init(&sha1_ctx);
sha1_update(&sha1_ctx, aes_ctx_decrypt.verify_2, 0x40u);
sha1_update(&sha1_ctx, &hash, 0x14u);
sha1_final(&sha1_ctx, &hash);
...
```

It should be noted that only a payload and a DWORD value that contains the package number are hashed. 4 DWORDs of the signature are not included in the hashed data.

The first 20 bytes of the received hash are compared with a package signature. If the match is found, the package is decrypted. If not, it is considered invalid.

Sending of the package to the server is performed in the same way.

The Trojan can execute three commands:

- Reverse Shell (cmd == 0x03)
- Download a file (cmd == 0x02)
- Upload a file to the command and control server (cmd == 0x01)

News about the Trojan