TigerRAT – Advanced Adversaries on the Prowl

blogs.vmware.com/security/2021/12/tigerrat-advanced-adversaries-on-the-prowl.html

December 3, 2021

Summary

On September 5th, 2021, the Korea Internet & Security Agency (KISA) released a report on a new threat they dubbed TigerRAT. The newly found malware shares similarities with malware previously reported by Kaspersky and Malwarebytes. Kaspersky has previously attributed those malware samples to Andariel, a threat actor group the Korean Financial Security Institute has identified as being a sub-group of Lazarus. TigerRAT appears to have been used from late 2020 onwards.

VMware's Threat Analysis Unit identifies TigerRAT as a payload associated with broader campaign of attacks against target enterprises. The TigerRAT payload capability includes the ability to manipulate files, execute remote commands, log keystrokes and remotely view and control the screen. TigerRAT may be blocked by VMware Carbon Black (see Figure 8).

Notably this malware, and the overall attack, originates from a loader application that utilizes a unique approach to storing the payload. Within the TigerRAT sample, configuration data for Command and Control (C2) communications is stored encrypted within the malware, and communications with the C2 server are customized to appear like HTTP web traffic.

VMware's Threat Analysis Unit performed a deep analysis of the TigerRAT malware to document its internal operations for comparison to other malware families in the wild.

When considering how TigerRAT may be used in the wild, defenders should recognize that the TigerRAT malware will be used by attackers as part of a broader campaign of attacks and that along the kill-chain, a wide variety of other malware types and attack techniques are also likely to be used. This is a key point when evaluating how such campaigns can be detected and disrupted.

Loader

Loader structure

In the case of sample , the TigerRAT payload data is stored in a section named "data". The payload structure is a 4-byte size, a 16-byte key, and then base64 encoded data. The size is the total length of the base64 data.

4 bytes	16 bytes	NN bytes
Size of base64 encoded data (NN)	Decryption key	base64 encoded data

The screenshot below (Figure 1) shows the size in red, the decryption key in green, and the base64 data in blue.

Figure 1: Loader

Loader function

The Loader’s purpose is to decrypt the final TigerRAT payload and execute it in memory. The data is loaded, base64 decoded and then the 16-byte key is used to decrypt the data with a simple XOR. The decoded payload is a PE file and after decrypting the loader will jump to the entry point.

Variance of loaders in the wild

The loader sample Malwarebytes reported on had almost identical code to that analyzed here, with the notable difference being that the base64 data was stored as overlay data after all the regular PE data. The embedded payload in the Malware bytes sample also differed. Refer to the Malwarebytes post for additional detail.

TigerRAT

The embedded PE is referred to as TigerRAT by the KISA report. A handful of different samples were found with compilation dates ranging from the end of 2020 to the beginning of 2021, with the only notable differences between different samples being the encrypted C2 information, and the DES/RC4 keys used for encryption and decryption.

The malware is written in C++ and makes use of only a handful of classes. At startup, a main class is created with references to the classes below.

Class name	Description
ProtocolTcpPure	Performs all the communication with the C2 server.
CryptorDES	Used to decrypt encrypted strings and data in the program.
CryptorRC4	Used to encrypt information sent to the C2 server and decrypt received commands.
IDGeneratorAdapter	Creates a unique ID for the victim machine used during C2 communication initialization.
ModuleUpdate	Handles C2 commands related to shutting down and self-deletion.
ModuleInformation	Handles C2 commands related to gathering victim machine information.
ModuleShell	Handles C2 commands related to executing commands.
ModuleFileManager	Handles C2 commands related to file manipulation and upload and download of files from the victim machine.
ModuleKeyLogger	Handles C2 commands related to starting and stopping keylogging functionality.
ModuleSocksTunnel	Handles C2 commands related to starting and stopping a socks tunnel.
ModuleScreenCapture	Handles C2 commands related to remote screen capturing and keyboard event injection.
ModulePortForwarder	Handles C2 commands related to starting and stopping port forwarding.

Table 1: TigerRAT classes

All of the Module classes inherit from a common base class and the main class stores an array of Module instances that are used during C2 communication. The code makes heavy use of threading when running actions based on C2 commands.

C2 Communication

During the main class initialization, the C2 IP addresses are decrypted using the CryptorDES class and stored in the main class. When that initialization is finished, the malware then attempts to initiate the network connection to the C2 server. The malware first tries to connect to one of the decrypted C2 IP addresses on port

443 and then performs a handshake with the C2 server. The malware starts by sending HTTP 1.1 /index.php?member=sbi2009 SSL3.3.7\x00 and then the C2 server responds with HTTP 1.1 200 OK SSL2.1\x00.

Following a successful initial handshake, the malware sends a 16-byte hash of the RC4 key being used and expects to get back a hardcoded 7-byte value. In the case of all currently found samples, the malware expects the 7-byte value “xPPygOn”.

The handshake process can be seen from the perspective of the C2 server by running a mockc2 TigerRAT server (Figure 2).

```
mockc2> debug on
[+] Debug output on
mockc2> listener start tigerrat 443
[DEBUG] Server listening
[i] connection from x.x.x.x:55067
[DEBUG] received

00000000  48 54 54 50 20 31 2e 31  20 2f 69 6e 64 65 78 2e  |HTTP 1.1 /index.|
00000010  70 68 70 3f 6d 65 6d 62  65 72 3d 73 62 69 32 30  |php?member=sbi20|
00000020  30 39 20 53 53 4c 33 2e  33 2e 37 00              |09 SSL3.3.7.|
[DEBUG] sent
00000000  48 54 54 50 20 31 2e 31  20 32 30 30 20 4f 4b 20  |HTTP 1.1 200 OK |
00000010  53 53 4c 32 2e 31 00                              |SSL2.1.|
[DEBUG] received
00000000  f2 7c 29 1f a5 75 fa 20  23 f7 7b 5b fa 5b e1 4a  |.|)..u. #.{[.[.J|
00000010  00                                                |.|
[DEBUG] sent
00000000  78 50 50 79 67 4f 6e 00                           |xPPygOn.|
```

Figure 2: TigerRAT handshake

After the handshake process has been completed successfully, the malware will proceed to send all further data in a standard command format and encrypted using the CryptorRC4 class. A single 32-byte RC4 key is used to initialize two separate running RC4 ciphers. One is used to decrypt incoming traffic and the other is used to encrypt outgoing traffic. The encrypted traffic has the following format (Figure 3):

```
struct packet {
uint32 size;
uint8 *data;
};
```

Figure 3: Encrypted traffic structure

Once decrypted the command format is as follows (Figure 4):

```
struct command {
uint32 module;
uint32 opcode;
```

```
uint32 size;
uint8 *data;
};
```

Figure 4: Command structure

After the handshake, the malware sends to the C2 server a unique victim machine identifier previously generated by the IDGeneratorAdapter class. The unique ID is generated by calling the GetAdaptersInfo API and getting the hardware address for one of the network devices on the victim machine (Figure 5).

```
[DEBUG] received
00000000  18 00 00 00 9d c6 28 3a  a8 14 21 6c 4f 27 81 0a  |……(:..!lO'..|
00000010  5c 4d 4d 42 cd 2e 65 fa  fd 50 b0 29             |\MMB..e..P.)|
[DEBUG] TigerRAT Command
[DEBUG] Module: 0x0
[DEBUG] Opcode: 0x1
[DEBUG]   Size: 0xc
[DEBUG]   Data:
00000000  f0 18 98 80 95 32 00 00  00 00 00 00             |…..2……|
```

Figure 5: TigerRAT victim ID

After the handshake process and upload of the victim ID, the malware initiates a heartbeat thread to send periodic packets to the C2 server, as well as a receive thread to read and process commands sent back from the C2 server. The subsequent actions of the malware will depend on the commands received from the C2 server; refer "Commands". An example of a heartbeat command can be seen below (Figure 6):

```
[DEBUG] received
00000000  0c 00 00 00 a5 31 6d a7  8f cd d4 70 aa e1 d4 56  |…..1m….p…V|
[DEBUG] TigerRAT Command
[DEBUG] Module: 0x0
[DEBUG] Opcode: 0x10
DEBUG]   Size: 0x0
```

Figure 6: TigerRAT heartbeat

Commands

Each Module class has a unique ID associated with it. This ID is set in the command structure sent from the C2 server down to the malware. The complete list of Module IDs can be seen below:

Module ID	Module Name
0x1	ModuleUpdate
0x2	ModuleInformation
0x3	ModuleShell
0x4	ModuleFileManager
0x5	ModuleKeyLogger

0x6	ModuleSocksTunnel
0x7	ModuleScreenCapture
0xa	ModulePortForwarder

Table 2: Module IDs

The following tables list the various opcodes used by the different Module classes and their function.

ModuleUpdate

Opcode	**Description**
0x20	Calls ExitProcess
0x30	Delete itself and exit

Table 3: ModuleUpdate opcodes

ModuleInformation

Opcode	**Description**
0x10	Retrieve victim's computer name using GetComputerNameW
0x20	Retrieve victim's Windows version using RtlGetVersion
0x30	Retrieve victiom's adapter info using GetAdaptersInfo
0x40	Retrieve victim's username using GetUserNameW

Table 4: ModuleInformation opcodes

ModuleShell

Opcode	**Description**
0x10	Execute a command
0x20	Set current directory
0x30	Get current directory
0x40	Test TCP connection

Table 5: ModuleShell opcodes

ModuleFileManager

Opcode	**Description**
0x10	Retrieve drive info

0x20	List files
0x30	Delete file
0x40	Start file upload to victim machine
0x42	Write data to uploaded file
0x43	Finish file upload to victim machine
0x50	Download file from victim machine
0x57	Set offset in file to download
0x5f	Wait for file transfers to finish
0x60	Call CreateProcessW
0x63	Call CreateProcessAsUserW
0x70	Download a directory from victim machine
0x80	Find files
0x90	Find files

Table 6: ModuleFileManager opcodes

ModuleKeyLogger

Opcode	Description
0x10	Initialize keylogger
0x11	Set keylogger flag
0x20	Stop keylogger
0x21	Set keylogger flag
0x25	Retrieve keylogger output
0x32	Retrieve keylogger file

Table 7: ModuleKeyLogger opcodes

ModuleSocksTunnel

Opcode	Description
0x10	Start socks tunnel
0x20	Forward data
0x30	Stop socks tunnel

Table 8: ModuleSocksTunnel opcodes

ModuleScreenCapture

Opcode	Description
0x10	Start screen capture
0x20	Stop screen capture
0x50	Modify mouse
0x52	Modify mouse
0x53	Modify mouse
0x60	Send VK_ESCAPE using keybd_event
0x61	Send VK_MENU + VK_TAB using keybd_event
0x62	Send VK_CONTROL + A using keybd_event
0x63	Send VK_RSHIFT + VK_DELETE using keybd_event
0x64	Send VK_MENU + VK_F4 using keybd_event
0x65	Send VK_RETURN using keybd_event
0x66	Send VK_SPACE using keybd_event
0x67	Send VK_TAB using keybd_event

Table 9: ModuleScreenCapture opcodes

ModulePortForwarder

Opcode	Description
0x11	Retrieve port forwarding status
0x20	Start port forwarding
0x30	Stop port forwarding

Table 10: ModulePortForwarder opcodes

Detection and Blocking

The TigerRAT malware may be detected . Figure 7 below shows TigerRAT launching multiple command interpreters in response to simulated commands sent from the mock C2 server. VMware Carbon Black Cloud can be configured to block unknown software attempting to run command interpreters as seen in Figure 8 below.

Figure 7: Process tree of TigerRAT executing remote commands

Figure 8: VMware Carbon Black Cloud blocking execution on unknown application attempting to run a command interpreter

MITRE ATT&CK TIDs

TID	Tactic	Description
T1059.003	Execution	Command and Scripting Interpreter: Windows Command Shell

T1134.002	Defense Evasion, Privilege Escalation	Access Token Manipulation: Create Process with Token
T1087.001	Discovery	Account Discovery: Local Account
T1083	Discovery	File and Directory Discovery
T1033	Discovery	System Owner/User Discovery
T1005	Collection	Data from Local System
T1056.001	Collection, Credential Access	Input Capture: Keylogging
T1113	Collection	Screen Capture
T1573.001	Command and Control	Encrypted Channel: Symmetric Cryptography
T1041	Exfiltration	Exfiltration Over C2 Channel

Indicators of Compromise (IOCs)

Indicator	Type	Context
1f8dcfaebbcd7e71c2872e0ba2fc6db81d651cf654a21d33c78eae6662e62392	SHA256	TigerRAT Loader
00331e5f972a98755811c02ec47301336a824a34	SHA1	TigerRAT Loader
4df757390adf71abdd084d3e9718c153	MD5	TigerRAT Loader
f32f6b229913d68daad937cc72a57aa45291a9d623109ed48938815aa7b6005c	SHA256	TigerRAT
b312dd587e8725edf782e0c176b902fbbfc01468	SHA1	TigerRAT
505262547f8879249794fc31eea41fc6	MD5	TigerRAT
29c6044d65af0073424ccc01abcb8411cbdc52720cac957a3012773c4380bab3	SHA256	TigerRAT
3d8bdbdc08b6cefc7a44c18fafe7e4032c3b68bf	SHA1	TigerRAT
a35a8c64870b9a3fe45348b4f2a93e75	MD5	TigerRAT
fed94f461145681dc9347b382497a72542424c64b6ae6fcf945f4becd2d46c32	SHA256	TigerRAT
e2f78ec89d80ed5c0299856fee84cc78c5d7f7ba	SHA1	TigerRAT
d6121d74dcef566a5e2f9aba179b8cca	MD5	TigerRAT
6dcfb2f52521672743f4888e992229896b98ab0e6bd979311ebdb4dcccc2b2e6	SHA256	TigerRAT
4a698b176e34d1c24c4fa13e9a773f90c6ce5413	SHA1	TigerRAT
2961c465a07bc80d206a09a6f5723a34	MD5	TigerRAT
ed11e94fd9aa3c7d4dd0b4345c106631fe52929c6e26a0daec2ed7d22e47ada0	SHA256	TigerRAT

0bced0f20ef12fbab59593dcd02e4c75d852b671	SHA1	TigerRAT
525cc10803d9858fca5dc4010925ba68	MD5	TigerRAT
52.202.193.124	TCP/443	TigerRAT C2
185.208.158.204	TCP/443	TigerRAT C2
185.208.158.208	TCP/443	TigerRAT C2