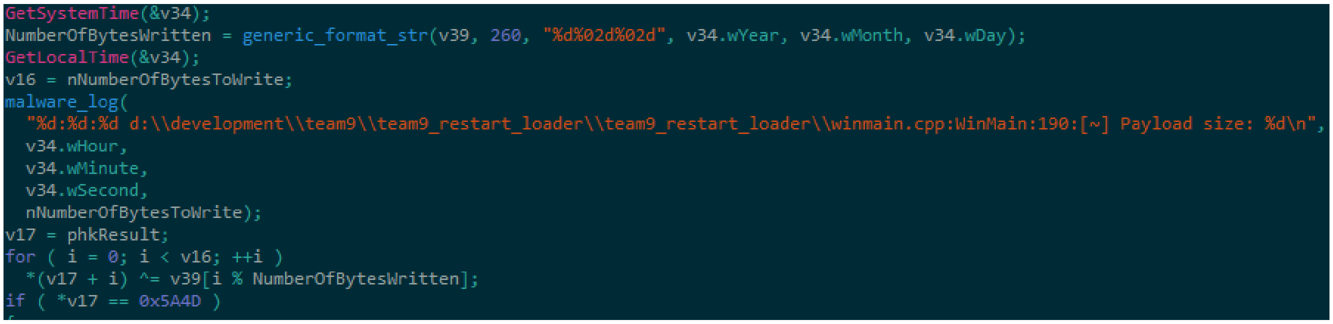
In-depth analysis of the new Team9 malware family

blog.fox-it.com/2020/06/02/in-depth-analysis-of-the-new-team9-malware-family/

June 2, 2020



```
GetSystemTime(&v34);
NumberOfBytesWritten = generic_format_str(v39, 260, "%d%02d%02d", v34.wYear, v34.wMonth, v34.wDay);
GetLocalTime(&v34);
v16 = nNumberOfBytesToWrite;
malware_log(
  "%d:%d:%d d:\\development\\team9\\team9_restart_loader\\team9_restart_loader\\winmain.cpp:WinMain:190:[~] Payload size: %d\n",
  v34.wHour,
  v34.wMinute,
  v34.wSecond,
  nNumberOfBytesToWrite);
v17 = phkResult;
for ( i = 0; i < v16; ++i )
  *(v17 + i) ^= v39[i % NumberOfBytesWritten];
if ( *v17 == 0x5A4D )
```

Author: Nikolaos Pantazopoulos
Co-author: Stefano Antenucci (@Antelox)
And in close collaboration with NCC's RIFT.

1. Introduction

Publicly discovered in late April 2020, the Team9 malware family (also known as 'Bazar [1]') appears to be a new malware being developed by the group behind Trickbot. Even though the development of the malware appears to be recent, the developers have already developed two components with rich functionality. The purpose of this blog post is to describe the functionality of the two components, the loader and the backdoor.

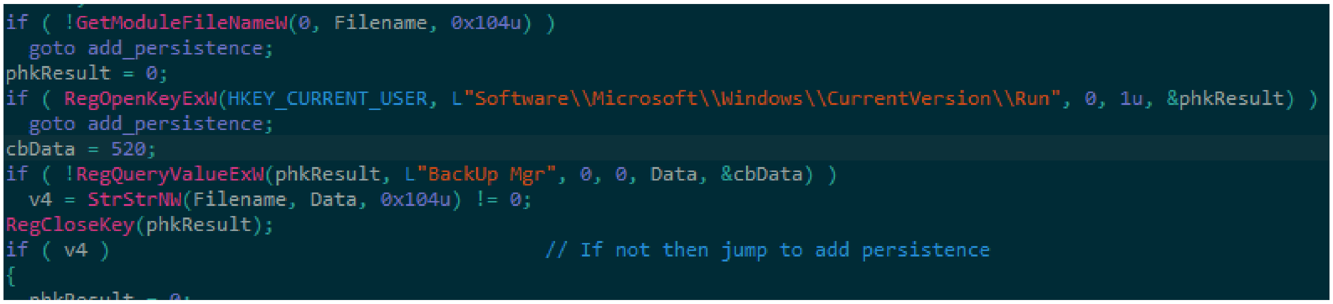
> *About the Research and Intelligence Fusion Team (RIFT):*
> RIFT leverages our strategic analysis, data science, and threat hunting capabilities to create actionable threat intelligence, ranging from IOCs and detection rules to strategic reports on tomorrow's threat landscape. Cyber security is an arms race where both attackers and defenders continually update and improve their tools and ways of working. To ensure that our managed services remain effective against the latest threats, NCC Group operates a Global Fusion Center with Fox-IT at its core. This multidisciplinary team converts our leading cyber threat intelligence into powerful detection strategies.

2. Early variant of Team9 loader

We assess that this is an earlier variant of the Team9 loader (35B3FE2331A4A7D83D203E75ECE5189B7D6D06AF4ABAC8906348C0720B6278A4) because of its simplicity and the compilation timestamp. The other variant was compiled more recently and has additional functionality. It should be noted that in very early versions of the loader binaries (2342C736572AB7448EF8DA2540CDBF0BAE72625E41DAB8FFF58866413854CA5C), the developers were using the Windows BITS functionality in order to download the backdoor. However, we believe that this functionality has been dropped.

Before proceeding to the technical analysis part, it is worth mentioning that the strings are not encrypted. Similarly, the majority of the Windows API functions are not loaded dynamically.

When the loader starts its execution, it checks if another instance of itself has infected the host already by attempting to read the value ‘BackUp Mgr’ in the ‘Run’ registry key ‘Software\Microsoft\Windows\CurrentVersion\Run’ (Figure 1). If it exists, it validates if the current loaders file path is the same as the one that has already been set in the registry value’s data (BackUp Mgr). Assuming that all of the above checks were successful, the loader proceeds to its core functionality.



```
if ( !GetModuleFileNameW(0, Filename, 0x104u) )
  goto add_persistence;
phkResult = 0;
if ( RegOpenKeyExW(HKEY_CURRENT_USER, L"Software\\Microsoft\\Windows\\CurrentVersion\\Run", 0, 1u, &phkResult) )
  goto add_persistence;
cbData = 520;
if ( !RegQueryValueExW(phkResult, L"BackUp Mgr", 0, 0, Data, &cbData) )
  v4 = StrStrNW(Filename, Data, 0x104u) != 0;
RegCloseKey(phkResult);
if ( v4 )                                // If not then jump to add persistence
{
  phkResult = 0;
```

Figure 1 – Loader verifies if it has already infect the host

However, if any of the above checks do not meet the requirements then the loader does one of the following actions:

1. Copy itself to the %APPDATA%\Microsoft folder, add this file path in the registry ‘Run’ key under the value ‘BackUp Mgr’ and then execute the loader from the copied location.
2. If the loader cannot access the %APPDATA% location or if the loader is running from this location already, then it adds the current file path in the ‘Run’ registry key under the value ‘BackUp Mgr’ and executes the loader again from this location.

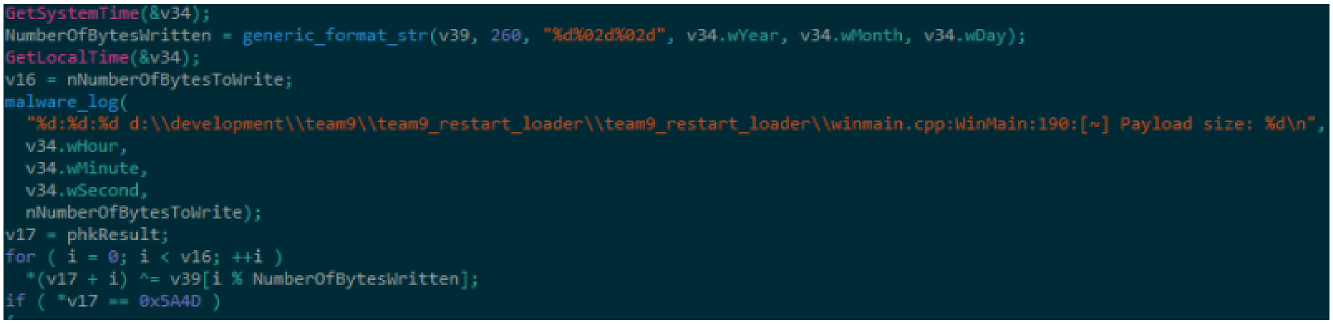
When the persistence operation finishes, the loader deletes itself by writing a batch file in the Windows temporary folder with the file name prefix ‘tmp’ followed by random digits. The batch file content:

```
@echo off
set Module=%1
:Repeat
del %Module%
if exist %Module% goto Repeat
del %0
```

Next, the loader fingerprints the Windows architecture. This is a crucial step because the loader needs to know what version of the backdoor to download (32-bit or 64-bit). Once the Windows architecture has been identified, the loader carries out the download.

The core functionality of the loader is to download the Team9 backdoor component. The loader contains two ‘.bazar’ top-level domains which point to the Team9 backdoor. Each domain hosts two versions of the Team9 backdoor on different URIs, one for each Windows architecture (32-bit and 64-bit), the use of two domains is highly likely to be a backup method.

Any received files from the command and control server are sent in an encrypted format. In order to decrypt a file, the loader uses a bitwise XOR decryption with the key being based on the infected host’s system time (Year/Month/Day) (Figure 2).



```
GetSystemTime(&v34);
NumberOfBytesWritten = generic_format_str(v39, 260, "%d%02d%02d", v34.wYear, v34.wMonth, v34.wDay);
GetLocalTime(&v34);
v16 = nNumberOfBytesToWrite;
malware_log(
  "%d:%d:%d d:\\development\\team9\\team9_restart_loader\\team9_restart_loader\\winmain.cpp:WinMain:190:[~] Payload size: %d\n",
  v34.wHour,
  v34.wMinute,
  v34.wSecond,
  nNumberOfBytesToWrite);
v17 = phkResult;
for ( i = 0; i < v16; ++i )
  *(v17 + i) ^= v39[i % NumberOfBytesWritten];
if ( *v17 == 0x5A4D )
```

Figure 2 – Generate XOR key based on infected host’s time

As a last step, the loader verifies that the executable file was decrypted successfully by validating the PE headers. If the Windows architecture is 32-bit, the loader injects the received executable file into ‘calc.exe’ (Windows calculator) using the ‘Process Hollowing’ technique. Otherwise, it writes the executable file to disk and executes it.

The following tables summarises the identified bazar domains and their URIs found in the early variants of the loader.

URI	Description
/api/v108	Possibly downloads the 64-bit version of the Team9 backdoor
/api/v107	Possibly downloads the 32-bit version of the Team9 backdoor
/api/v5	Possibly downloads an updated 32-bit version of the Team9 loader
/api/v6	Possibly downloads an updated 64-bit version of the Team9 loader
/api/v7	Possibly downloads the 32-bit version of the Team9 backdoor
/api/v8	Possibly downloads the 64-bit version of the Team9 backdoor

Table 1 – Bazar URIs found in early variants of the loader

The table below (table 2) summarises the identified domains found in the early variants of the loader.

Bazar domains
bestgame[.]bazar
forgame[.]bazar
zirabuo[.]bazar
tallcareful[.]bazar
coastdeny[.]bazar

Table 2 – Bazar domains found in early variants of the loader

Lastly, another interesting observation is the log functionality in the binary file that reveals the following project file path:

d:\\development\\team9\\team9_restart_loader\\team9_restart_loader

3. Latest variant of Team9 loader

In this section, we describe the functionality of a second loader that we believe to be the latest variant of the aforementioned Team9 loader. This assessment is based on three factors:

1. Similar URIs in the backdoor requests
2. Similar payload decryption technique
3. Similar code blocks

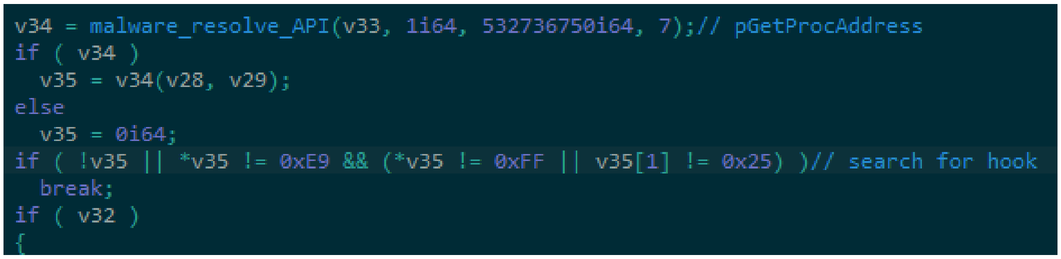
Unlike its previous version, the strings are encrypted and the majority of Windows API functions are loaded dynamically by using the Windows API hashing technique.

Once executed, the loader uses a timer in order to delay the execution. This is likely used as an anti-sandbox method. After the delayed time has passed, the loader starts executing its core functionality.

Before the malware starts interacting with the command and control server, it ensures that any other related files produced by a previous instance of the loader will not cause any issues. As a result the loader appends the string '_lyrt' to its current file path and deletes any file with this name. Next, the loader searches for the parameter '-p' in the command line and if found, it deletes the scheduled task 'StartDT'. The loader creates this scheduled task later for persistence during execution. The loader also attempts to execute hijacked shortcut files, which will eventually execute an instance of Team9 loader. This functionality is described later.

The loader performs a last check to ensure that the operating systems keyboard and language settings are not set to Russian and creates a mutex with a hardcoded name 'ld_201127'. The latter is to avoid double execution of its own instance.

As mentioned previously, the majority of Windows API functions are loaded dynamically. However, in an attempt to bypass any API hooks set by security products, the loader manually loads 'ntdll' from disk, reads the opcodes from each API function and compares them with the ones in memory (Figure 3). If the opcodes are different, the loader assumes a hook has been applied and removes it. This applies only to 64-bit samples reviewed to date.



```
v34 = malware_resolve_API(v33, 1i64, 532736750i64, 7);// pGetProcAddress
if ( v34 )
  v35 = v34(v28, v29);
else
  v35 = 0i64;
if ( !v35 || *v35 != 0xE9 && (*v35 != 0xFF || v35[1] != 0x25) )// search for hook
  break;
if ( v32 )
{
```

Figure 3 – Scan for hooks in Windows API functions

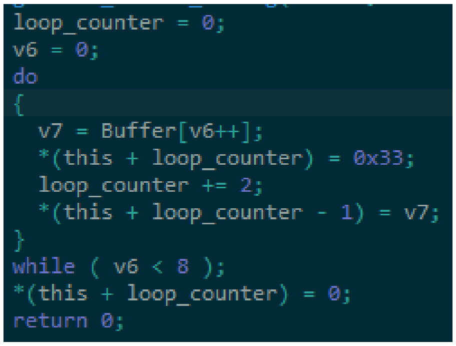
The next stage downloads from the command and control server either the backdoor or an updated version of the loader. It is interesting to note that there are minor differences in the loader's execution based on the identified Windows architecture and if the '-p' parameter has been passed into the command line.

Assuming that the ‘-p’ parameter has not been passed into the command line, the loader has two loops. One for 32-bit and the other for 64-bit, which download an updated version of the loader. The main difference between the two loops is that in case of a Windows x64 infection, there is no check of the loader’s version.

The download process is the same with the previous variant, the loader resolves the command and control server IP address using a hardcoded list of DNS servers and then downloads the corresponding file. An interesting addition, in the latest samples, is the use of an alternative command and control server IP address, in case the primary one fails. The alternative IP address is generated by applying a bitwise XOR operation to each byte of the resolved command and control IP address with the byte 0xFE. In addition, as a possible anti-behaviour method, the loader verifies that the command and control server IP address is not ‘127.0.0.1’. Both of these methods are also present in the latest Team9 backdoor variants.

As with the previous Team9 loader variant, the command and control server sends back the binary files in an encrypted format. The decryption process is similar with its previous variant but with a minor change in the XOR key generation, the character ‘3’ is added between each hex digit of the day format (Figure 4). For example:

```
3323303323303303353313 38 (ASCII format, host date: 2020-05-18)
```



```
loop_counter = 0;
v6 = 0;
do
{
  v7 = Buffer[v6++];
  *(this + loop_counter) = 0x33;
  loop_counter += 2;
  *(this + loop_counter - 1) = v7;
}
while ( v6 < 8 );
*(this + loop_counter) = 0;
return 0;
```

Figure 4 – Add the character ‘3’ in the generated XOR key

If the ‘-p’ parameter has been passed into the command line, the loader proceeds to download the Team9 backdoor directly from the command and control server. One notable addition is the process injection (hollow process injection) when the backdoor has been successfully downloaded and decrypted. The loader injects the backdoor to one of the following processes:

1. Svchost
2. Explorer
3. cmd

Whenever a binary file is successfully downloaded and properly decrypted, the loader adds or updates its persistence in the infected host. The persistence methods are available in table 3.

Persistence Method	Persistence Method Description
Scheduled task	The loader creates two scheduled tasks, one for the updated loader (if any) and one for the downloaded backdoor. The scheduled task names and timers are different.

Winlogon hijack	Add the malware’s file path in the ‘Userinit’ registry value. As a result, whenever the user logs in the malware is also executed.
Shortcut in the Startup folder	The loaders creates a shortcut, which points to the malware file, in the Startup folder. The name of the shortcut is ‘adobe’.
Hijack already existing shortcuts	The loader searches for shortcut files in Desktop and its subfolders. If it finds one then it copies the malware into the shortcut’s target location with the application’s file name and appends the string ‘__’ at the end of the original binary file name. Furthermore, the loader creates a ‘.bin’ file which stores the file path, file location and parameters. The ‘.bin’ file structure can be found in the Appendix section. When this structure is filled in with all required information, It is encrypted with the XOR key 0x61.

Table 3 – Persistence methods loader

The following tables summarises the identified bazar domains and their URIs for this Team9 loader variant.

URI	Description
/api/v117	Possibly downloads the 32-bit version of the Team9 loader
/api/v118	Possibly downloads the 64-bit version of the Team9 loader
/api/v119	Possibly downloads the 32-bit version of the Team9 backdoor
/api/v120	Possibly downloads the 64-bit version of the Team9 backdoor
/api/v85	Possibly downloads the 32-bit version of the Team9 loader
/api/v86	Possibly downloads the 64-bit version of the Team9 loader
/api/v87	Possibly downloads the 32-bit version of the Team9 backdoor
/api/v88	Possibly downloads the 64-bit version of the Team9 backdoor

Table 4 – Identified URIs for Team9 loader variant

Bazar domain
bestgame[.]bazar
forgame[.]bazar

Table 5 – Identified domains for Team9 loader variant

4. Team9 backdoor

We are confident that this is the backdoor which the loader installs onto the compromised host. In addition, we believe that the first variants of the Team9 backdoor started appearing in the wild in late March 2020. Each variant does not appear to have major changes and the core of the backdoor remains the same.

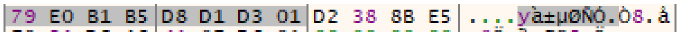
During analysis, we identified the following similarities between the backdoor and its loader:

1. Creates a mutex with a hardcoded name in order to avoid multiple instances running at the same time (So far the mutex names which we have identified are 'mn_185445' and '{589b7a4a-3776-4e82-8e7d-435471a6c03c}')
2. Verifies that the keyboard and the operating system language is not Russian
3. Use of Emercoin domains with a similarity in the domain name choice

Furthermore, the backdoor generates a unique ID for the infected host. The process that it follows is:

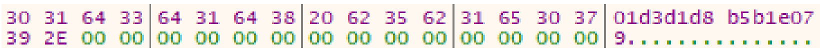
1. Find the creation date of 'C:\Windows' (Windows FILETIME structure format). The result is then converted from a hex format to an ASCII representation. An example is shown in figures 5 (before conversion) and 6 (after conversion).
2. Repeat the same process but for the folder 'C:\Windows\System32'
3. Append the second string to the first with a bullet point as a delimiter. For example, 01d3d1d8 b10c2916.01d3d1d8 b5b1e079
4. Get the NETBIOS name and append it to the previous string from step 3 along with a bullet point as a delimiter. For example: 01d3d1d8 b10c2916.01d3d1d8 b5b1e079.DESKTOP-4123EEB.
5. Read the volume serial number of C: drive and append it to the previous string. For example: 01d3d1d8 b10c2916.01d3d1d8 b5b1e079.DESKTOP-SKCF8VA.609fbbd5
6. Hash the string from step 5 using the MD5 algorithm. The output hash is the bot ID.

Note: In a few samples, the above algorithm is different. The developers use hard-coded dates, the Windows directory file paths in a string format ('C:\Windows' and 'C:\Windows\system32') and the NETBIOS name. Based on the samples' functionality, there are many indications that these binary files were created for debugging purposes.



```
79 E0 B1 B5|D8 D1 D3 01|D2 38 8B E5|....yà±µØÑÓ.Ò8.å
```

Figure 5 – Before conversion



```
30 31 64 33|64 31 64 38|20 62 35 62|31 65 30 37|01d3d1d8 b5b1e07
39 2E 00 00|00 00 00 00|00 00 00 00|00 00 00 00|9...............
```

Figure 6 – After conversion

4.1 Network communication

The backdoor appears to support network communication over ports 80 (HTTP) and 443(HTTPS). In recent samples, a certificate is issued from the infected host for communication over HTTPS. Each request to the command and control server includes at least the following information:

1. A URI path for requesting tasks (/2) or sending results (/3).
2. Group ID. This is added in the 'Cookie' header.

Lastly, unlike the loader which decrypts received network replies from the command and control server using the host's date as the key, the Team9 backdoor uses the bot ID as the key.

4.2 Bot commands

The backdoor supports a variety of commands. These are summarised in the table below.

Command ID	Description	Parameters
0	Set delay time for the command and control server requests	Time to delay the requests
1	Collect infected host information	Memory buffer to fill in the collected data
10	Download file from an address and inject into a process using either hollowing process injection or Doppelgänging process injection	• DWORD value that represents the corresponding execution method. This includes: ◦ Process hollowing injection ◦ Process Doppelgänging injection ◦ Write the file into disk and execute it • Process mask – DWORD value that represents the process name to inject the payload. This can be one of the following: 1. Explorer 2. Cmd 3. Calc (Not used in all variants) 4. Svchost 5. notepad • Address from which the file is downloaded • Command line
11	Download a DLL file and execute it	• Timeout value • Address to download the DLL • Command line • Timeout time
12	Execute a batch file received from the command and control server	• DWORD value to determine if the batch script is to be stored into a Windows pipe (run from memory) or in a file into disk • Timeout value. • Batch file content
13	Execute a PowerShell script received from the command and control server	• DWORD value to determine if the PowerShell script is to be stored into a Windows pipe (run from memory) or in a file into disk • Timeout value. • PowerShell script content
14	Reports back to the command and control server and terminates any handled tasks	None

15	Terminate a process	PID of the process to terminate
16	Upload a file to the command and control server. Note: Each variant of the backdoor has a set file size they can handle.	Path of the file to read and upload to the command and control server.
100	Remove itself	None

Table 6 – Supported backdoor commands

Table 7 summarises the report structure of each command when it reports back (POST request) to the command and control server. Note: In a few samples, the backdoor reports the results to an additional IP address (185.64.106[.]73) If it cannot communicate with the Bazar domains.

Command ID/Description	Command execution results structure
1/ Collect infected host information	The POST request includes the following information: • Operating system information • Operating system architecture • NETBIOS name of the infected host • Username of the infected user • Backdoor’s file path • Infected host time zone • Processes list • Keyboard language • Antivirus name and installed applications • Infected host’s external IP • Shared drives • Shared drives in the domain • Trust domains • Infected host administrators • Domain admins
11/ Download a DLL file and execute it	The POST request includes the following parameters: • Command execution errors (Passed in the parameter ‘err’) • Process identifier (Passed in the ‘pid’ parameter) • Command execution output (Passed in the parameter ‘stdout’, if any) • Additional information from the command execution (Passed in the parameter ‘msg’, if any)
12/ Execute a batch file received from the command and control server	Same as the previous command (11/ Download a DLL file and execute it)
13/ Execute a PowerShell script received from the command and control server	Same as the previous command (11/ Download a DLL file and execute it)

14/ Reports back to the command and control server and terminate any handled tasks	POST request with the string ‘ok’
15/ Terminate a process	Same as the previous command (11/ Download a DLL file and execute it)
16/ Upload a file to the command and control server	No parameters. The file’s content is sent in a POST request.
100/ Remove itself	POST request with the string ‘ok’ or ‘process termination error’

Table 7 – Report structure

5. Appendix

5.1 struct shortcut_bin

```
struct shortcut_bin

{
BYTE junk_data[434];
BYTE file_path[520];
BYTE filepath_dir[520];
BYTE file_loader_parameters[1024];
};
```

5.2 IOCs

File hashes

Description	SHA-256 Hash
Team9 backdoor (x64)	4F258184D5462F64C3A752EC25FB5C193352C34206022C0755E48774592B7707
Team9 backdoor (x64)	B10DCEC77E00B1F9B1F2E8E327A536987CA84BCB6B0C7327C292F87ED603837D
Team9 backdoor (x64)	363B6E0BC8873A6A522FE9485C7D8B4CBCFFA1DA61787930341F94557487C5A8
Team9 backdoor (x64)	F4A5FE23E21B6B7D63FA2D2C96A4BC4A34B40FD40A921B237A50A5976FE16001
Team9 backdoor (x64)	A0D0CFA8BF0BC5B8F769D8B64EAB22D308B108DD8A4D59872946D69C3F8C58A5

Team9 backdoor (x64)	059519E03772D6EEEA9498625AE8B8B7CF2F01FC8179CA5D33D6BCF29D07C9F4
Team9 backdoor (x64)	0F94B77892F22D0A0E7095B985F30B5EDBE17AB5B8D41F798EF0C708709636F4
Team9 backdoor (x64)	2F0F0956628D7787C62F892E1BD9EDDA8B4C478CF8F1E65851052C7AD493DC28
Team9 backdoor (x64)	37D713860D529CBE4EAB958419FFD7EBB3DC53BB6909F8BD360ADAA84700FAF2
Team9 backdoor (x64)	3400A7DF9EC3DC8283D5AC7ACCB6935691E93FEDA066CC46C6C04D67F7F87B2B
Team9 backdoor (x64)	5974D938BC3BBFC69F68C979A6DC9C412970FC527500735385C33377AB30373A
Team9 backdoor (x64)	C55F8979995DF82555D66F6B197B0FBCB8FE30B431FF9760DEAE6927A584B9E3
Team9 backdoor (x86)	94DCAA51E792D1FA266CAE508C2C62A2CA45B94E2FDFBCA7EA126B6CD7BC5B21
Team9 backdoor (x86)	4EE0857D475E67945AF2C5E04BE4DEC3D6D3EB7C78700F007A7FF6F8C14D4CB3
Team9 backdoor (x86)	8F552E9CA2BEDD90CE9935A665758D5DE2E86B6FDA32D98918534A8A5881F91A
Team9 backdoor (x86)	AE7DAA7CE3188CCFE4069BA14C486631EEA9505B7A107A17DDEE29061B0EDE99
Team9 backdoor (x86)	F3C6D7309F00CC7009BEA4BE6128F0AF2EA6B87AB7A687D14092F85CCD35C1F5
Team9 backdoor (x86)	6CBF7795618FB5472C5277000D1C1DE92B77724D77873B88AF3819E431251F00
Team9 backdoor (x86)	B0B758E680E652144A78A7DDECC027D4868C1DC3D8D7D611EC4D3798358B0CE5

Team9 backdoor (x86)	959BA7923992386ABF2E27357164672F29AAC17DDD4EE1A8AD4C691A1C566568
Team9 backdoor (x86)	3FE61D87C9454554B0CE9101F95E18ABAD8AC6C62DCC88DC651DDFB20568E060
Team9 loader (x64)	B3764EF42D526A1AE1A4C3B0FE198F35C6BC5C07D5F155D15060B94F8F6DC695
Team9 loader (x64)	210C51AAB6FC6C52326ECE9DBD3DDAB5F58E98432EF70C46936672C79542FBD0
Team9 loader (x64)	11B5ADAEFD04FFDACEB9539F95647B1F51AEC2117D71ECE061F15A2621F1ECE9
Team9 loader (x64)	534D60392E0202B24D3FDAF992F299EF1AF1FB5EFEF0096DD835FE5C4E30B0FA
Team9 loader (x64)	9D3A265688C1A098DD37FE77C139442A8EB02011DA81972CEDDC0CF4730F67CF
Team9 loader (x64)	CE478FDBD03573076394AC0275F0F7027F44A62A306E378FE52BEB0658D0B273
Team9 loader (x64)	5A888D05804D06190F7FC408BEDE9DA0423678C8F6ECA37ECCE83791DE4DF83D
Team9 loader (x64)	EB62AD35C613A73B0BD28C1779ACE80E2BA587A7F8DBFEC16CF5BF520CAA71EE
Team9 loader (x64)	A76426E269A2DEFABCF7AEF9486FF521C6110B64952267CFE3B77039D1414A41
Team9 loader (x64)	65CDBDD03391744BE87AC8189E6CD105485AB754FED0B069A1378DCA3E819F28
Team9 loader (x64)	38C9C3800DEA2761B7FAEC078E4BBD2794B93A251513B3F683AE166D7F186D19
Team9 loader (x64)	8F8673E6C6353187DBB460088ADC3099C2F35AD868966B257AFA1DF782E48875
Team9 loader (x86)	35B3FE2331A4A7D83D203E75ECE5189B7D6D06AF4ABAC8906348C0720B6278A4
Team9 loader (x86)	65E44FC8527204E88E38AB320B3E82694D1548639565FDAEE53B7E0F963D3A92
Team9 loader (x86)	F53509AF91159C3432C6FAF4B4BE2AE741A20ADA05406F9D4E9DDBD48C91EBF9
Team9 loader (x86)	73339C130BB0FAAD27C852F925AA1A487EADF45DF667DB543F913DB73080CD5D
Team9 loader (x86)	2342C736572AB7448EF8DA2540CDBF0BAE72625E41DAB8FFF58866413854CA5C

Team9 loader (x86)	079A99B696CC984375D7A3228232C44153A167C1936C604ED553AC7BE91DD982
Team9 loader (x86)	0D8AEACF4EBF227BA7412F8F057A8CDDC54021846092B635C8D674B2E28052C6
Team9 loader (x86)	F83A815CE0457B50321706957C23CE8875318CFE5A6F983A0D0C580EBE359295
Team9 loader (x86)	3FA209CD62BACC0C2737A832E5F0D5FD1D874BE94A206A29B3A10FA60CEB187D
Team9 loader (x86)	05ABD7F33DE873E9630F9E4F02DBD0CBC16DD254F305FC8F636DAFBA02A549B3

Table 8 – File hashes

Identified Emercoin domains

Domains
newgame[.]bazar
thegame[.]bazar
portgame[.]bazar
workrepair[.]bazar
realfish[.]bazar
eventmoult[.]bazar
bestgame[.]bazar
forgame[.]bazar
Zirabuo[.]bazar

Table 9 – Identified Emercoin domains

Command and Control IPs

C&C IPs
34.222.222[.]126
71.191.52[.]192
77.213.120[.]90
179[.]43.134.164
185[.]65.202.62
220[.]32.32.128

34[.]222.222.126
51[.]81.113.26
71[.]191.52.192
77[.]213.120.90
85[.]204.116.58

Table 10 – Command and Control IPs

Identified DNS IPs

DNS IPs
51[.]254.25.115
193[.]183.98.66
91[.]217.137.37
87[.]98.175.85
185[.]121.177.177
169[.]239.202.202
198[.]251.90.143
5[.]132.191.104
111[.]67.20.8
163[.]53.248.170
142[.]4.204.111
142[.]4.205.47
158[.]69.239.167
104[.]37.195.178
192[.]99.85.244
158[.]69.160.164
46[.]28.207.199
31[.]171.251.118
81[.]2.241.148
82[.]141.39.32
50[.]3.82.215

46[.]101.70.183
5[.]45.97.127
130[.]255.78.223
144[.]76.133.38
139[.]59.208.246
172[.]104.136.243
45[.]71.112.70
163[.]172.185.51
5[.]135.183.146
51[.]255.48.78
188[.]165.200.156
147[.]135.185.78
92[.]222.97.145
51[.]255.211.146
159[.]89.249.249
104[.]238.186.189
139[.]59.23.241
94[.]177.171.127
45[.]63.124.65
212[.]24.98.54
178[.]17.170.179
185[.]208.208.141
82[.]196.9.45
146[.]185.176.36
89[.]35.39.64
89[.]18.27.167
77[.]73.68.161
185[.]117.154.144
176[.]126.70.119

139[.]99.96.146
217[.]12.210.54
185[.]164.136.225
192[.]52.166.110
63[.]231.92.27
66[.]70.211.246
96[.]47.228.108
45[.]32.160.206
128[.]52.130.209
35[.]196.105.24
172[.]98.193.42
162[.]248.241.94
107[.]172.42.186
167[.]99.153.82
138[.]197.25.214
69[.]164.196.21
94[.]247.43.254
94[.]16.114.254
151[.]80.222.79
176[.]9.37.132
192[.]71.245.208
195[.]10.195.195

Table 11 – Identified DNS IPs

Mutexes

Component	Mutex name
Team9 backdoor	mn_185445
Team9 backdoor	{589b7a4a-3776-4e82-8e7d-435471a6c03c}
Team9 loader	ld_201127

Table 12 – Mutex names Team9 components

Host IOCs

1. Files ending with the string ‘_lyrt’
2. Scheduled tasks with names ‘StartAT’ and ‘StartDT’
3. Shortcut with file name ‘adobe’ in the Windows ‘StartUp’ folder
4. Registry value name ‘BackUp Mgr’ in the ‘Run’ registry key

Network detection

```
alert dns $HOME_NET any -> any 53 (msg:”FOX-SRT – Suspicious – Team9 Emercoin DNS Query
Observed”; dns_query; content:”.bazar”; nocase; dns_query;
pcre:”/(newgame|thegame|portgame|workrepair|realfish|eventmoult|bestgame|forgame|zirabuo)\.bazar/i”;
threshold:type limit, track by_src, count 1, seconds 3600; classtype:trojan-activity; metadata:created_at
2020-05-28; metadata:ids suricata; sid:21003029; rev:3;)
```

Source(s):

[1] https://www.bleepingcomputer.com/news/security/bazarbackdoor-trickbot-gang-s-new-stealthy-network-hacking-malware/