Taiwan Heist: Lazarus Tools and Ransomware

Archived: 2026-04-05 12:44:19 UTC

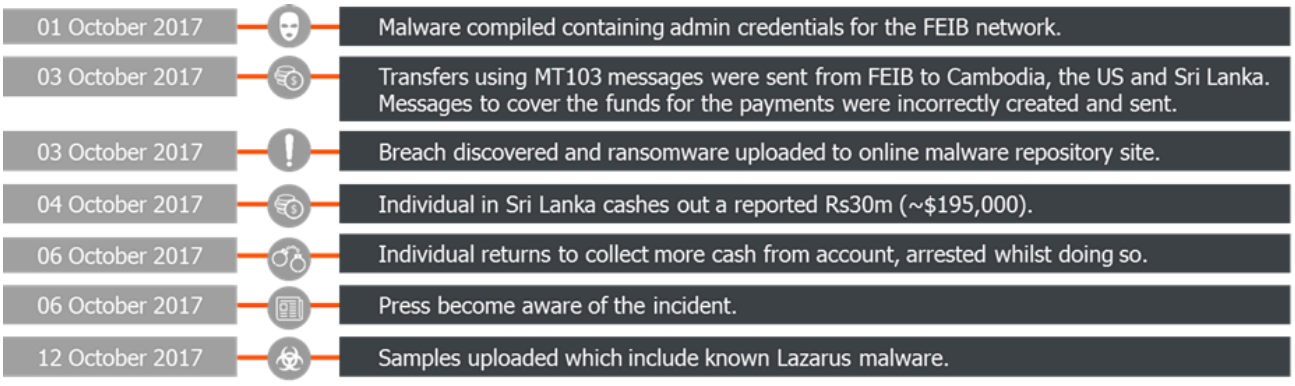
Written by Sergei Shevchenko, Hirman Muhammad bin Abu Bakar, and James Wong

BACKGROUND

Reports emerged just over a week ago of a new cyber-enabled bank heist in Asia. Attackers targeting Far Eastern International Bank (FEIB), a commercial firm in Taiwan, moved funds from its accounts to multiple overseas beneficiaries. In a story which reminds us of the Bangladesh Bank case – the culprits had compromised the bank's system connected to the SWIFT network and used this to perform the transfers.

In recent days, various malware samples have been uploaded to malware repositories which appear to originate from the intrusion. These include both known Lazarus group tools, as well as a rare ransomware variant called 'Hermes' which may have been used as a distraction or cover-up for the security team whilst the heist was occurring.

The timeline below provides an overview of the key events:



Little information is available at present about when or how the attackers compromised the bank, but it is likely more details will emerge in the coming weeks. This blogpost seeks to summarise what is in the public domain at the moment, as well as analyse the samples uploaded to malware repositories.

ANALYSIS

Several files have been uploaded to malware databases which appear to be related to this attack, including an archive titled "FEIB_Samples" submitted from Taiwan on 12th Oct 2017. These and other samples are listed below:

#	MD5	Filenames	Submitted From	First Seen	Compile Time
1	`9563e2f443c3b4e1b00f25be0a30d56e`	`FEIB_Samples_pwd(Virus).zi_`	TW	2017-10-12 02:50:16	N/A

2	`d08f1211fe0138134e822e31a47ec5d4`	`bitsran.exe`	TW	2017-10-03 01:01:31	2017-10-01 15:37:31
3	`b27881f59c8d8cc529fa80a58709db36`	`RSW7B37.tmp`	-	2017-10-03 01:01:37	2017-10-01 11:34:07
4	`3c9e71400b72cc0213c9c3e4ab4df9df`	`msmpeng.exe`	US	2017-10-07 08:58:00	2017-02-20 11:09:30
5	`0edbad9e6041d43f97c7369439a40138`	`FileTokenBroker.dll`	TW	2017-10-12 02:50:15	2017-01-05 01:11:33
6	`97aaf130cfa251e5207ea74b2558293d`	`splwow32.exe`	TW	2017-10-12 02:50:15	2017-02-20 11:09:30
7	`62217af0299d6e241778adb849fd2823`	N/A	GB	2017-10-08 03:32:47	2017-09-21 09:27:43
8	`0dd7da89b7d1fe97e669f8b4156067c8`	N/A	MY	2017-03-14 02:13:01	2017-03-06 17:32:58
9	`61075faba222f97d3367866793f0907b`	N/A	MY	2017-02-16 03:25:00	2017-02-10 15:03:30

File #1 is the ZIP file containing samples #2-6 inside. Samples #2-4 were also separately uploaded by users in Taiwan and the US on the dates given above.

Samples #7-9 are older versions of the Hermes ransomware.

Malware Analysis – Sample #2; Bitsran loader / spreader

Sample #2 is designed to run and spread a malicious payload on the victim's network. On execution, the malware places a copy of itself into the location:

`C:\Windows\Temp\bitsran.exe`

Next, the file establishes a persistence mechanism with the registry key:

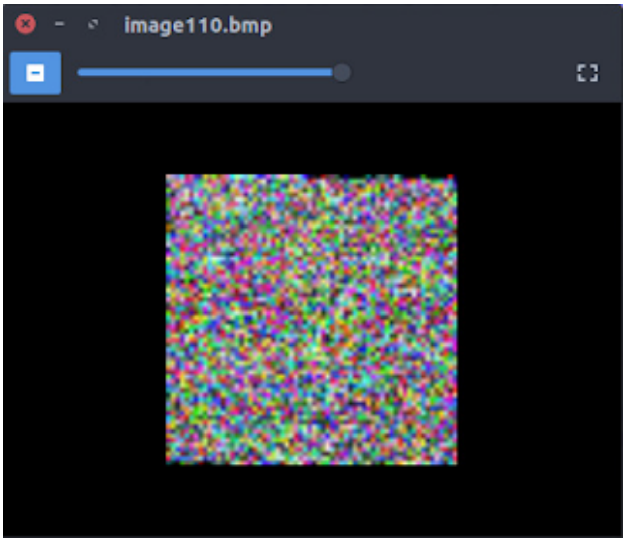
`HKLM\Software\Microsoft\Windows\CurrentVersion\Run`

It sets the value of `'BITSRAN'` to point to the executable in the `Temp` location above.

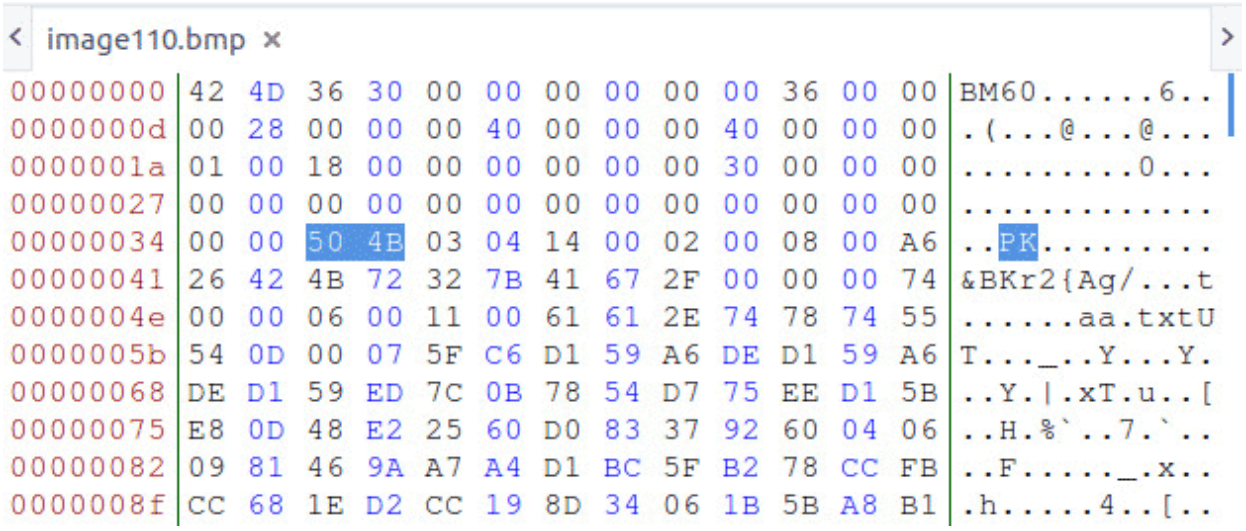
The malware then enumerates all processes, searching for specific anti-virus processes and attempts to kill these using the command line tool `taskkill`.

Process Name	Process Description
`tmbmsrv.exe`	Trend Micro Unauthorized Change Prevention Service
`tmccsf.exe`	Trend Micro OfficeScan Common Client Solution Framework
`cntaosmgr.exe`	Trend Micro OfficeScan Add-on Service Client Management Service
`ntrtscan.exe`	Trend Micro OfficeScan NT RealTime Scan
`pccntmon.exe`	Trend Micro OfficeScan Antivirus real-time scan monitor
`tmlisten.exe`	Trend Micro OfficeScan NT Listener
`tmpfw.exe`	Trend Micro OfficeScan NT Firewall

Next, the process attempts to find an embedded `'IMAGE'` resource with offset `#110`. If successful, this file is loaded into memory. When manually extracting this file, it can be seen to represent a pixelated bitmap (BMP) file.



However, further investigation reveals that the file is what is known as a 'Polyglot' file, whereby a file is contained within another file. Using a HEX viewer, it is possible to see that this file also contains a ZIP file (beginning at the 'PK' header), with the pixelated image above referencing the bytes of the file to be RGB values.



The contents of this resource is decompressed from offset 54, with the last 4 bytes of the file specifying the ZIP's file size in bytes. When successfully decrypted, the file is saved into the same directory as the initial executable. This takes the filename `'RSWXXXX.tmp'`, where `'XXXX'` is randomly generated through the GetTempFileName function. Once written to disk, this process is created through the CreateProcess function. Sample #3 (`RSW7B37.tmp`) is an example of this file.

Whilst this additional payload is executing, the initial malware attempts to copy itself to other devices on the network. Two user accounts are hardcoded into the malware, and are used to establish connections to the `C$` SMB shares on Windows devices. These are the accounts:

Account Name	Account Password
`FEIB\SPUSER14`	`#ED{REMOVED}`
`FEIB\scomadmin`	`!it{REMOVED}`

Both accounts clearly relate to FEIB, though we couldn't confirm whether the credentials are valid or not. The `SPUSER14` may be a Sharepoint user account whilst `scomadmin` likely corresponds to System Center Operations Manager admin – an account for managing machines in a data centre.

Instead of enumerating all devices on the network, the malware iterates through a hardcoded list of 5357 IP addresses, in the ranges:

- 10.49.*
- 10.50.*
- 10.51.*
- 10.59.*

It is assumed that previous reconnaissance was conducted by the actors on the internal network to identify active and responding devices, as well as capturing admin credentials for the network.

If a device successfully responds to a SMB packet on port 445, the malware copies itself to the `C$` network share using the provided credentials, writing the file to the location:

`C:\Windows\Temp\bitsran.exe`

If successful, a further command is executed using the same credentials, to create a scheduled task on the remote device with the name 'BITSRAN'. The full command executed is:

`cmd.exe /c schtasks /create /tn "BITSRAN" /tr /s /u /p /st 00:00 /et 23:59 /sc minute /mo 1 /ru system /f`

Malware Analysis – Sample #3, Dropped file / Hermes Ransomware

The dropped file is a variant of the Hermes ransomware.

The ransomware calls *GetSystemDefaultLangID()* to obtain language identifier for the system locale. It contains a list of three system language codes: `0x0419` (Russian), `0x0422` (Ukrainian), and `0x0423` (Belarusian). However, it only checks against the last two, and, if matching, the malware quits. Whether this is a false-flag or not is unknown.

The ransomware deletes the Volume Shadow Copies (a type of backup on Windows), using command:

`vssadmin Delete Shadows /all /quiet`

Following that, it deletes all VSS (Volume Shadow Copy Service) backup files (which include System Restore files) and orphaned shadows, by running commands below for the drives from `C:`, `D:`, `E:`, `F:`, `G:`, and `H:`

`vssadmin resize shadowstorage /for=%DRIVE% /on=%DRIVE% /maxsize=401MB vssadmin resize shadowstorage /for=%DRIVE% /on=%DRIVE% /maxsize=unbounded`

The trick above is called *"pulling the carpet"* as it forces Windows to voluntarily dump all shadows due to lack of space.

The ransomware then recursively deletes all backup files from the drives `C:`, `D:`, `E:`, `F:`, `G:`, and `H:`, having the following extensions:

- `*.VHD`
- `*.bac`
- `*.bak`
- `*.wbcat`
- `*.bkf`
- `Backup*.*`
- `backup*.*`
- `*.set`
- `*.win`
- `*.dsk`

Using Windows CryptoAPI platform, the malware creates an exchange key pair, and then exports the 2,048-bit public RSA key into an external file called `PUBLIC`.

The ransomware then enumerates both local and network resources, and encrypts files using AES256 algorithm.

Each encrypted directory will have a ransom note left in it:

```
HERMES 2.1 RANSOMWARE radical edition
All your important files are encrypted
Your files has been encrypted using RSA2048 algorithm with unique public-key stored on your PC.
There is only one way to get your files back: contact with us, pay, and get decryptor software.
You have "UNIQUE_ID_DO_NOT_REMOVE" file on your desktop also it duplicated in some folders,
its your unique idkey, attach it to letter when contact with us. Also you can decrypt 3 files for test.
We accept Bitcoin, you can find exchangers on https://www.bitcoin.com/buy-bitcoin and others.
Contact information: BM-2cVcZL1xfve1yGGKwEBgG1ge6xJ5PYGfGw@bitmessage.ch
reserve: BM-2cT4U1vBdjfqKDeWMEXgCWs9SfnMK1GLTF@bitmessage.ch
```

Malware Analysis – Samples #4 and #6, Lazarus malware

Sample #4 (`msmpeng.exe`) is packed with Themida to hamper analysis under a debugger, a monitoring application, or a virtual machine.

Once fully unpacked in memory, it appears to be an x86 variant of the `fdsvc.dll` backdoor described in our February blogpost "Lazarus' False Flag Malware". This malware was discovered on networks in Poland and Mexico, following a series of watering-hole attacks.

Just like before, the backdoor uses several transliterated Russian words to either indicate the state of its communication or issue backdoor commands:

State/Command	Translation from Russian	Meaning
`Nachalo`	beginning	start communication session
`ustanavlivat`	to set	handshake state
`poluchit`	to receive	receive data
`pereslat`	to send	send data
`derzhat`	to maintain	maintain communication session
`vykhodit`	to exit	exit communication session
`kliyent2podklyuchit`	client to connect	client is ready to connect

Sample #6 (`splwow32.exe`) is the same backdoor, only it's not packed.

Both sample #4 and #6 have the same time stamp: 20 February 2017, 11:09:30. It appears that sample #6 was actually obtained by packing sample #4 with Themida (potentially, to avoid detection), as code/data found in both samples is identical.

The backdoor expects a command line parameter that specifies remote C&C address and port number. If it is executed with no command-line parameters, it quits.

The specified command-line parameter is decrypted, using some basic character manipulations and applying XOR with 2 keys:

```
0x517A4563 ( “QzEc” )
0x77506F66 ( “wPof” )
```

The decrypted string is expected to delimit C&C address and port number with the `“:”` character. Multiple C&Cs can be delimited with the `“|”` character.

If the backdoor finds no valid pair of C&C address and port number delimited with the `“:”` character, it quits.

Otherwise, it starts polling the remote C&C for a remote task to execute. Each polling attempt starts from a state `“Nachalo”` (“start communication session”), with a 3 second delay between each attempt to connect to the C&C.

Each connection attempt starts from a state called `“kliyent2podklyuchit”` (“client is ready to connect”).

If the backdoor fails to connect five times, or if it connects, but the task it receives is `“vykhodit”` (“exit communication session”), then the backdoor will quit. Otherwise, it will execute the remote command, effectively giving the attackers full control over the compromised system. After the execution, the polling cycle continues.

Malware Analysis – Sample #5

`FileTokenBroker.dll` is a DLL, installed as a service under the `svchost.exe` (`netsvcs`) service host.

Once loaded as a service DLL, the DLL's export *ServiceMain()* is called. The DLL then constructs a file name that consists of the host process name, formatted as:

```
%SYSTEM%\en-US\[HOST_PROCESS_NAME_NO_EXTENSION].dll.mui
```

For example, if the DLL is loaded into the address space of `svchost.exe` , the constructed filename will be:

```
c:\windows\system32\en-US\svchost.dll.mui
```

Another possible name is:

```
c:\windows\system32\en-US\netsvc.dll.mui
```

The DLL then reads this file, and decrypts it with a running XOR mask. Once decrypted, it further reads an RC4 key from it, and decrypts it with the RC4 algorithm.

The decrypted file will contain a hash, so the DLL checks the hash as well to make sure the integrity of the decrypted file is intact.

A fully decrypted file is then parsed as a PE file, and loaded as a DLL.

Hence, `FileTokenBroker.dll` decrypts and executes a payload that is created by an external dropper or is implanted by the attackers.

The `%SYSTEM%\en-US` directory will have multiple system files in it, so it is chosen to blend the encrypted payload file with the other legitimate system files. Unlike other `*.dll.mui` files in `%SYSTEM%\en-US` directory that are MZ files, the encrypted payload is not an MZ file.

Malware Analysis – Samples #7, #8, and #9, Further Hermes malware

Samples #7, #8, and #9 relate to previous instances of Hermes ransomware.

Malware of this category is typically widespread, but in the case of Hermes it seems relatively rare. This is suspicious in itself and reminds us of WannaCry – another rarely observed ransomware. Further analysis is on-going to understand the history of this malware variant.

Transactions

Through working with trusted partners, we have been able to get insight into the transactions made as part of the heist. The transactions consisted of two common SWIFT message types, `MT103` and `MT202COV`.

`MT103` messages are used for normal, cross border, cash transfers which would typically request funds be transferred into a personal or company beneficiary account. `MT103` messages can be used on their own, or can be coupled with a cover message; `MT202COV` is used to order the movement of funds to the beneficiary institution via another financial institution/Intermediary Bank.

In this heist the attackers created `MT103` messages to transfer funds to Cambodia, the US, and Sri Lanka. In addition to the `MT103` messages, the attackers created `MT202COV` messages; the content of these messages was syntactically correct but the values in specific fields were wrong. As a result, they were received by the intermediary bank but had no further influence on the funds transferred to the beneficiary accounts.

Reports of $60M being stolen appear to be due to confusion over these latter messages, and the amounts actually stolen were considerably lower. Most of these appear to have been recovered.

Further details of the destination accounts within Sri Lanka have emerged in open source. The money had been transferred to the Bank of Ceylon in Sri Lanka on 3 October. The following day, an individual in Sri Lanka allegedly withdrew RS 30m (about $195K). Two days after that, the same individual returned to withdraw a further RS 8m, but was arrested when he arrived at the bank. Sri Lankan police have since arrested another individual and a further suspect is wanted by Sri Lankan law enforcement.

CONCLUSIONS

It has been over a year since the last activity on a payments system from the attackers behind the infamous Bangladesh Bank heist. Lazarus, the prime suspects, have been busy nonetheless – targeting Bitcoin in various ways, as well as other intrusions into banks such as in Poland and Mexico (albeit without evidence of targeting payment systems). In one of these cases we and other researchers were able to observe infrastructure in North Korea controlling the malware – further clues as to the origins of these attackers.

The attack this month on Taiwanese Far Eastern International Bank has some of the hallmarks of the Lazarus group:

- • Destination beneficiary accounts in Sri Lanka and Cambodia – both countries have been used previously as destinations for Lazarus' bank heist activity;
- • Use of malware previously seen in Lazarus' Poland and Mexico bank attacks. Where these files were found and the context of their use needs to be confirmed, but could provide a crucial attributive link;
- • Use of unusual ransomware, potentially as a distraction.

Despite their continued success in getting onto payment systems in banks, the Lazarus group still struggle getting the cash in the end, with payments being reversed soon after the attacks are uncovered. The group may be trying new tricks to disrupt victims and delay their ability to respond – such as different message formats, and the deployment of ransomware across the victim's network as a smokescreen for their other activity. It's likely they'll continue their heist attempts against banks in the coming months and we expect they will evolve their modus operandi to incorporate new ways of disrupting victims (and possibly the wider community) from responding.

More work needs to be done to identify how FEIB was attacked, whether further custom tools were involved, confirm the context of the Lazarus malware in the intrusion, and where else this Hermes ransomware has been seen.

Assuming Lazarus are indeed back to targeting bank payment systems, this will serve to emphasize the importance of network hardening and controls frameworks being pushed by the industry at present.

RECOMMENDATIONS

Some general network hardening and monitoring lessons can be taken from this:

- • Firewall off SMB (445) for internal computers. If access to this service is required, it should be permitted only for those IP's that require access. i.e. 445 is required for SCOM to push an agent install, therefore 445 should only be allowed from that source server;
- • Application blacklisting should be implemented to prevent the use of tools such as `vssadmin.exe`, `cmd.exe`, `powershell.exe` and similar;
- • File Integrity Monitoring should be considered and configured to monitor file creations in "trusted" locations such as the System32 directory. This can also be used to monitor deletes, with an alert configured to fire on excessive deletes in a row;
- • Windows Security Event logs should be monitored to capture Scheduled Task creation events – Event ID `4698`;
- • Registry Auditing should be enabled and monitored to capture any additions to `HKLM\Software\Microsoft\Windows\CurrentVersion\Run`;
- • Excessive use of known administrative privilege accounts should be alerted on – specifically in a "one to many" behavioural configuration. i.e. is one specific IP connecting to a large number of devices using the same credentials in a short period of time;
- • Ensure privileged accounts have a complex password that does not include any part of the username, or application it relates to.

Additional longer term recommendations for financial institutions:

- • Practice incident response scenarios which include complex attacks combining covert payment fraud and overt network disruption through ransomware, DDoS, network downtime, etc.

- • Ensure that you are progressing towards being able to attest against the SWIFT 27 controls.
  For more information see:
  http://www.baesystems.com/en/cybersecurity/swift-customer-security-programme

APPENDIX A – INDICATORS OF ATTACK

MD5 Hashes	`d08f1211fe0138134e822e31a47ec5d4`
	`b27881f59c8d8cc529fa80a58709db36`
	`3c9e71400b72cc0213c9c3e4ab4df9df`
	`0edbad9e6041d43f97c7369439a40138`
	`97aaf130cfa251e5207ea74b2558293d`
	`62217af0299d6e241778adb849fd2823`
	`0dd7da89b7d1fe97e669f8b4156067c8`
	`61075faba222f97d3367866793f0907b`
File / Process name	`bitsran.exe`

APPENDIX B – YARA RULE

```
rule Hermes2_1 {
  meta:
     date = "2017/10/11"
     author = "BAE"
     hash = "b27881f59c8d8cc529fa80a58709db36"
  strings:
     $magic = { 4D 5A }
     //in both version 2.1 and sample in Feb
     $s1 = "SYSTEM\\CurrentControlSet\\Control\\Nls\\Language\\"
     $s2 = "0419"
     $s3 = "0422"
     $s4 = "0423"
     //in version 2.1 only
     $S1 = "HERMES"
     $S2 = "vssadminn"
     $S3 = "finish work"
```

```
      $S4 = "testlib.dll"
      $S5 = "shadowstorageiet"
      //maybe unique in the file
      $u1 = "ALKnvfoi4tbmiom3t40iomfr0i3t4jmvri3tb4mvi3btv3rgt4t777"
      $u2 = "HERMES 2.1 TEST BUILD, press ok"
      $u3 = "hnKwtMcOadHwnXutKHqPvpgfysFXfAFTcaDHNdCnktA" //RSA Key part
   condition:
      $magic at 0 and all of ($s*) and 3 of ($S*) and 1 of ($u*)
}
```

Source: https://baesystemsai.blogspot.com/2017/10/taiwan-heist-lazarus-tools.html