

# THE REGIN PLATFORM NATION-STATE OWNAGE OF GSM NETWORKS

Kaspersky Lab Report

Version 1.0  
24 November 2014

## Contents

|                                                                     |    |
|---------------------------------------------------------------------|----|
| Introduction, history .....                                         | 3  |
| Initial compromise and lateral movement .....                       | 3  |
| The Regin platform .....                                            | 4  |
| Stage 1 – 32/64 bit .....                                           | 4  |
| Stage 2 – loader – 32-bit .....                                     | 7  |
| Stage 2 – loader – 64-bit .....                                     | 8  |
| Stage 3 – 32-bit – kernel mode manager “VMEM.sys” .....             | 8  |
| Stage 3 – 64-bit .....                                              | 9  |
| Stage 4 (32-bit) / 3 (64-bit) – dispatcher module, ‘disp.dll’ ..... | 9  |
| 32-bit .....                                                        | 9  |
| 64-bit .....                                                        | 9  |
| Stage 4 – Virtual File Systems (32/64-bit) .....                    | 10 |
| Unusual modules and artifacts .....                                 | 16 |
| Artifacts .....                                                     | 16 |
| GSM targeting .....                                                 | 18 |
| Communication and C&C .....                                         | 20 |
| Victim statistics .....                                             | 22 |
| Attribution .....                                                   | 23 |
| Conclusions .....                                                   | 23 |
| Technical appendix and indicators of compromise .....               | 24 |
| Yara rules .....                                                    | 24 |
| MD5s .....                                                          | 25 |
| Registry branches used to store malware stages 2 and 3 .....        | 26 |
| C&C IPs .....                                                       | 26 |
| VFS RC5 decryption algorithm .....                                  | 27 |

*"Beware of Regin, the master! His heart is poisoned. He would be thy bane..." -  
"The Story of Siegfried" by James Baldwin*

## Introduction, history

In the spring of 2012, following a Kaspersky Lab presentation on the unusual facts surrounding the Duqu malware ([http://www.kaspersky.com/about/press/major\\_malware\\_outbreaks/duqu](http://www.kaspersky.com/about/press/major_malware_outbreaks/duqu)), a security researcher contacted us and mentioned that Duqu reminded him of another high-end malware incident. Although he couldn't share a sample, the researcher mentioned 'Regin', a type of malware attack that is now dreaded by security administrators in many government agencies around the world.

For the past three years we have been tracking this most elusive malware all around the world. From time to time samples would appear on various multi-scanner services, but they were all unrelated to each other, cryptic in functionality, and lacking in context.

It is unknown exactly when the first samples of Regin appeared in the wild. Some of them have timestamps dating back to 2003.

The victims of Regin fall into the following categories:

- Telecom operators
- Government institutions
- Multinational political bodies
- Financial institutions
- Research institutions
- Individuals involved in advanced mathematical/cryptographic research

So far, we've observed two main objectives of the attackers:

- Intelligence gathering
- Facilitating other types of attacks

While in most cases the attackers were focused on extracting sensitive information such as emails and other electronic documents, we have observed cases where the attackers compromised telecom operators to enable the launch of additional sophisticated attacks. This is discussed in detail in the **GSM attacks** section, below.

Perhaps one of the most well-known victims of Regin was Jean Jacques Quisquater ([https://en.wikipedia.org/wiki/Jean-Jacques\\_Quisquater](https://en.wikipedia.org/wiki/Jean-Jacques_Quisquater)), a well-known Belgian cryptographer. In February 2014, Quisquater announced he was the victim of a sophisticated cyber-intrusion incident. We were able to obtain samples from the Quisquater case and confirm they belong to the Regin platform.

Another victim of Regin was a computer we call the '**Magnet of Threats**'. The computer belongs to a certain research institution and, besides Regin, it has been attacked by **Animal Farm**, **Itaduke**, **Mask/Careto**, **Turla**, and some other advanced threats that do not have public names, all co-existing happily on the same computer at some point.

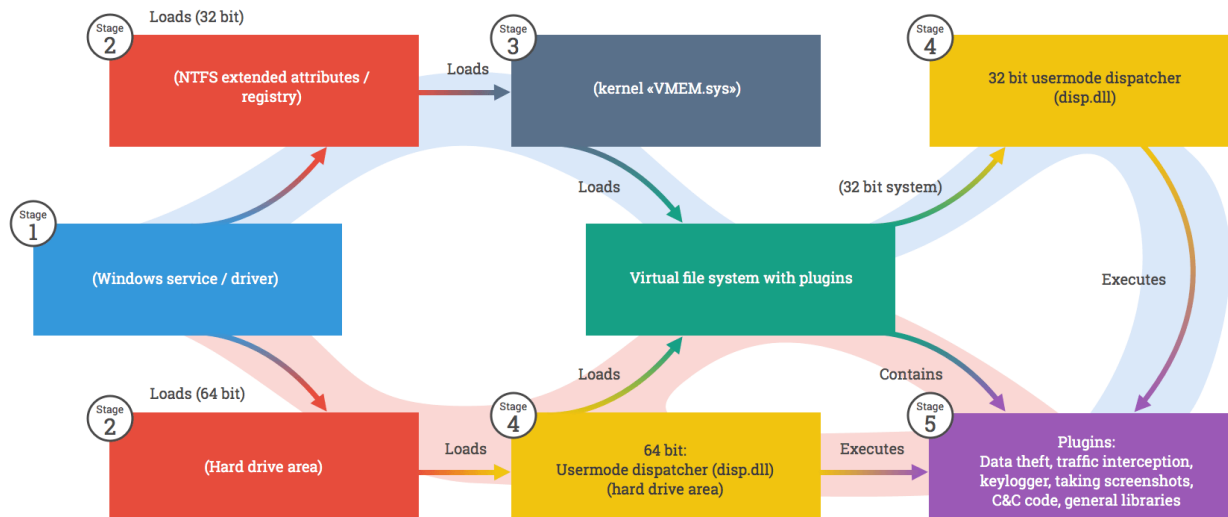
## Initial compromise and lateral movement

The exact method used for the initial compromise remains a mystery, although several theories exist, including use of man-in-the-middle attacks with browser zero-day exploits. For some of the victims we observed tools and modules designed for lateral movement. So far we have not encountered any exploits. The replication modules are copied to remote computers using Windows administrative shares and then executed. Obviously this technique requires administrative privileges inside the victim's network. In several cases the infected machines were also Windows domain controllers. Targeting of system administrators via web-based exploits is a simple way of achieving immediate administrative access to the entire network.

## The Regin platform

Although some private research groups refer to it as the 'Regin malware', it is not entirely accurate to use the term malware in this case. In essence, Regin is a cyberattack *platform*, which the attackers deploy in victim networks for total remote control at *all levels*.

The platform is extremely modular in nature and has multiple stages.



GREAT KASPERSKY

Regin platform diagram

### Stage 1 – 32/64 bit

| Known MD5s:                      |
|----------------------------------|
| 01c2f321b6bfdb9473c079b0797567ba |
| 06665b96e293b23acc80451abb413e50 |
| 187044596bc1328efa0ed636d8aa4a5c |
| 1c024e599ac055312a4ab75b3950040a |
| 26297dc3cd0b688de3b846983c5385e5 |
| 2c8b9d2885543d7ade3cae98225e263b |
| 47d0e8f9d7a6429920329207a32ecc2e |
| 4b6b86c7fec1c574706cecedf44abded |
| 6662c390b2bbbd291ec7987388fc75d7 |
| 744c07e886497f7b68f6f7fe57b7ab54 |
| b269894f434657db2b15949641a67532 |
| b29ca4f22ae7b7b25f79c1d4a421139d |
| b505d65721bb2453d5039a389113b566 |
| ba7bb65634ce1e30c1e5415be3d1db1d |
| bfbe8c3ee78750c3a520480700e440f8 |
| d240f06e98c8d3e647cbf4d442d79475 |
| db405ad775ac887a337b02ea8b07fddc |
| ffb0b9b5b610191051a7bdf0806e1e47 |

In general, the first samples victims detect in their networks are stage 1 loaders. These are the easiest to notice because they are the only executables that exist directly on the victim's computer.

These samples use an odd technique to load the next stages, which until recently was unique to Regin. Interestingly, in mid-2012, the ZeroAccess gang implemented a very similar loading mechanism, which possibly suggests it learned about Regin and its unique features. (See <http://www.symantec.com/connect/blogs/trojanzeroaccess-sc-hidden-ntfs-ea>).

The particular feature used (or abused) by Regin to hide its next stages is called NTFS Extended Attributes (EA). Originally, these were implemented in Windows NT for compatibility with OS/2 applications; however, they made their way into later versions of Windows, namely 2000, XP and Vista. The malware hides its modules in NTFS EAs, splitting large files into several blocks of limited size. These are dynamically joined, decrypted and executed in memory.

Most of the stage 1 samples we have seen appear to have been built on top of other source code projects, which are 'piggybacked'; for instance, the Ser8UART project:  
<http://www.mirror-service.org/sites/downloads.sourceforge.net/s/se/ser8uart-driver/ser8uart-driver/Ser8UART%20%201.1.2.1/>.

For instance, the Regin loader with md5 **01c2f321b6bfdb9473c079b0797567ba** was built on top of the Ser8UART source code. A careful examination however spots the encrypted configuration block at offset 0x5600.

We can assume the attackers take various low-level open-source projects or Windows DDK source codes and merge them together with their malicious loader. Hence, each stage 1 loader looks very different from others, as it contains random useless code from various other programs. This technique makes it more difficult to build reliable detection for the loaders.

Despite the differences, all stage 1 samples are similar in functionality. They contain an encrypted config block that points to the next stages:

|                               |                         |                     |
|-------------------------------|-------------------------|---------------------|
| 2600: 09 DB EF 23 27 11 AF 00 | 01 6F 43 52 AE 6B 4B 6A | o n# ' « » @oCR«kKj |
| 2610: F2 62 45 52 80 49 78 5F | C6 46 5D 54 80 41 7C 4B | ≥bERCIx _ F]TCA K   |
| 2620: DD 77 4A 7A 83 48 65 48 | CC 50 5E 59 BD 70 57 72 | WJzâHeH P^Y pWr     |
| 2630: D3 7E 4D 63 82 65 5E 62 | FA 6C 4D 7B 9A 7E 49 7D | l~Mcée^b·lM{Ü~I}    |
| 2640: E3 57 52 1B 97 28 10 43 | AE 43 64 7A 90 56 6D 4C | πWR«û(«C«CdzÉVml    |
| 2650: CD 4F 6E 67 96 47 12 41 | D1 50 6C 03 89 40 78 5A | =OngûG†A_TPlvë@xZ   |
| 2660: C9 5A 7E 0C F0 4F 78 1B | F8 53 71 77 89 42 78 5E | ƒZ~♀≡Ox«°SqwëBx^    |
| 2670: C0 5B 79 7F 81 4A 70 56 | C8 43 61 67 99 52 68 4E | [yoüJpVlCagÖRhN     |
| 2680: D0 4B 69 6F 91 5A 60 46 | D8 B3 91 97 69 A2 98 BE | lKioæZ`F† æüiöyJ    |
| 2690: 20 BB 99 9F 61 AA 90 B6 | 28 A3 81 87 79 B2 88 AE | ηÖfa-É   (üücy  ë«  |
| 26A0: 30 AB 89 8F 71 BA 80 A6 | 38 93 B1 B7 49 82 B8 9E | 0%ëAq  C^8ô  Ié7R   |
| 26B0: 00 9B B9 BF 41 8A B0 96 | 08 83 A1 A7 59 92 A8 8E | ††Aè:ÜQâiYÆ;Ä       |
| 26C0: 10 8B A9 AF 51 9A A0 86 | 18 F3 D1 D7 29 E2 D8 FE | ►i-»QÜää†≤† )††     |
| 26D0: 60 B8 B5 BE 52 99 D0 F6 | 68 E3 C1 C7 39 F2 C8 EE | ˘†††RÖ  ÷hπ  9≥L€   |
| 26E0: 70 EB C9 CF 31 FA C0 E6 | 78 D3 F1 F7 09 C2 F8 DE | pδ††1·Lµx  ±≈O_T°   |
| 26F0: 40 DB F9 FF 01 CA F0 D6 | 48 C3 E1 E7 19 D2 E8 CE | @†·Θ  ≡†H†B†↓††0†   |
| 2700: 50 CB E9 D3 46 93 AE 82 | 17 64 42 29 E9 22 18 3E | P††0lFô«ë†dB)0†↑>   |

Once decrypted, the block contains several folder names and registry key names:

```
\REGISTRY\Machine\System\CurrentControlSet\Control\Class\{4F20E605-9452-4787-B793-D0204917CA58}
Class
<WINDOWS>
<WINDOWS>\fonts
Decrypted configuration
```

In the example above, the stage 1 tries to load a second stage from the extended attributes of the system directory specified in the configuration block (in our case, the WINDOWS folder). It also tries to read additional data from the EAs of the second directory (in our case, the WINDOWS\fonts directory). The second attribute value is optional and may have been used to overcome size limitations.

If the first EA data block is missing, the module also tries to read the complete body of the 2nd stage from a registry value using the key and value names from the configuration block.

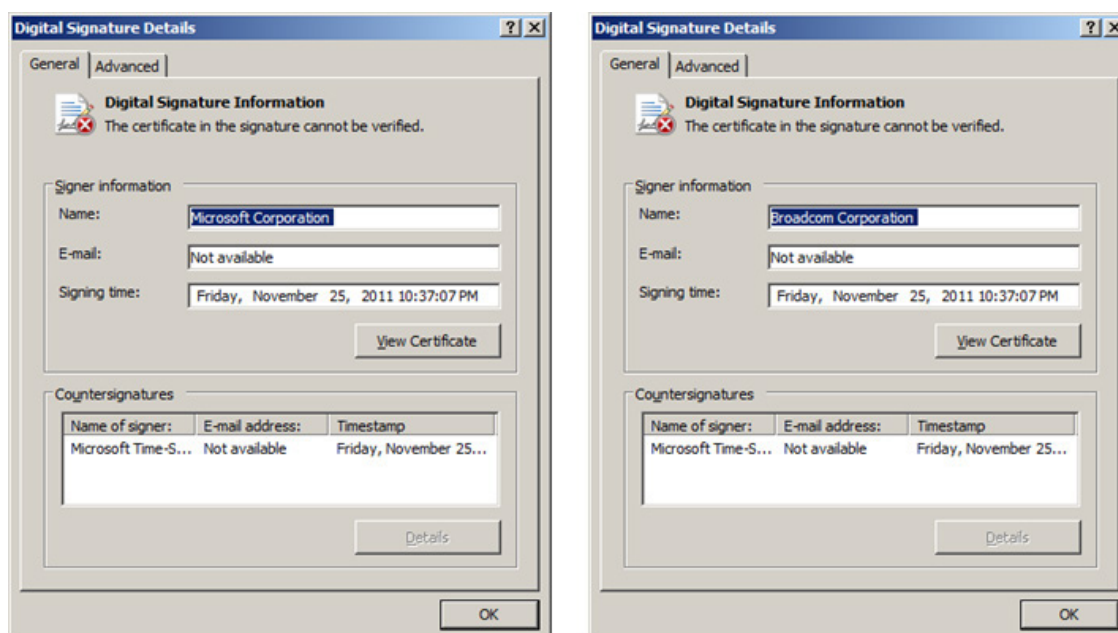
The body of the second stage is encrypted with one of two algorithms that are simple variations of XOR, and is supposed to be a PE file. The first stage loads that file in memory and calls its entry point function.

The 64-bit variant works in a slightly different way. Instead of storing the 2nd stage in the registry or extended attributes, the attackers preferred to store it after the end of the last partition on disk.

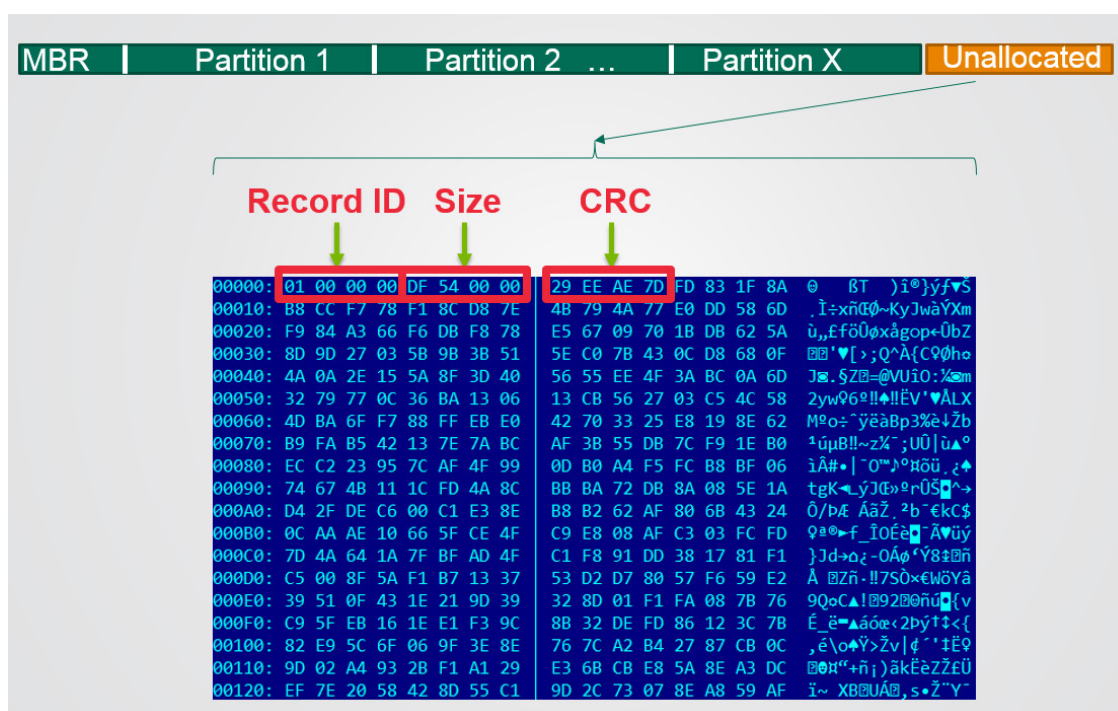
Known filenames for the 64-bit stage 1:

- system32\wsharp.dll – detected on a victim machine in Germany
- system32\wshnetc.dll – detected on a victim machine in Belgium

All the stage 1 modules for 64-bit systems were signed with fake digital certificates. The two fake certificates we identified are supposed to belong to Microsoft Corporation and Broadcom Corporation. During the infection phase, the attackers inject a trusted CA in the certificates chain, which instructs the system to trust their signatures.



Here is what the hard drive of a 64-bit system infected with Regin looks like:



Interestingly, while the 32-bit Regin stage 1 runs in kernel mode, on 64-bit systems the attacker code starts in user mode. This is perhaps due to the fact that it is more difficult to run kernel mode on modern Windows 64-bit systems.

## Stage 2 – loader – 32-bit

| Known MD5s:                      |
|----------------------------------|
| 18d4898d82fcb290dfed2a9f70d66833 |
| b9e4f9d32ce59e7c4daf6b237c330e25 |

The second stage for 32-bit systems is implemented as a driver module. It has a configuration block encrypted in a similar way to the first stage module. The configuration block contains the names of two system directories that hold the encrypted third stage in their extended attributes. It also has the name of a registry value that may hold the body of the third stage in case the EAs are missing (for computers with a FAT/FAT32-formatted system disk).

Once the encrypted third stage is read from the registry or NTFS EAs, it is decrypted using the RC5 algorithm and a fixed 16-byte key that is hardcoded in the second stage. Then, it is decompressed using the NR2e algorithm from the open-source UCL library. The second stage module loads the resulting binary in memory, validates that it is a valid PE file, and calls its entry point in a system thread.

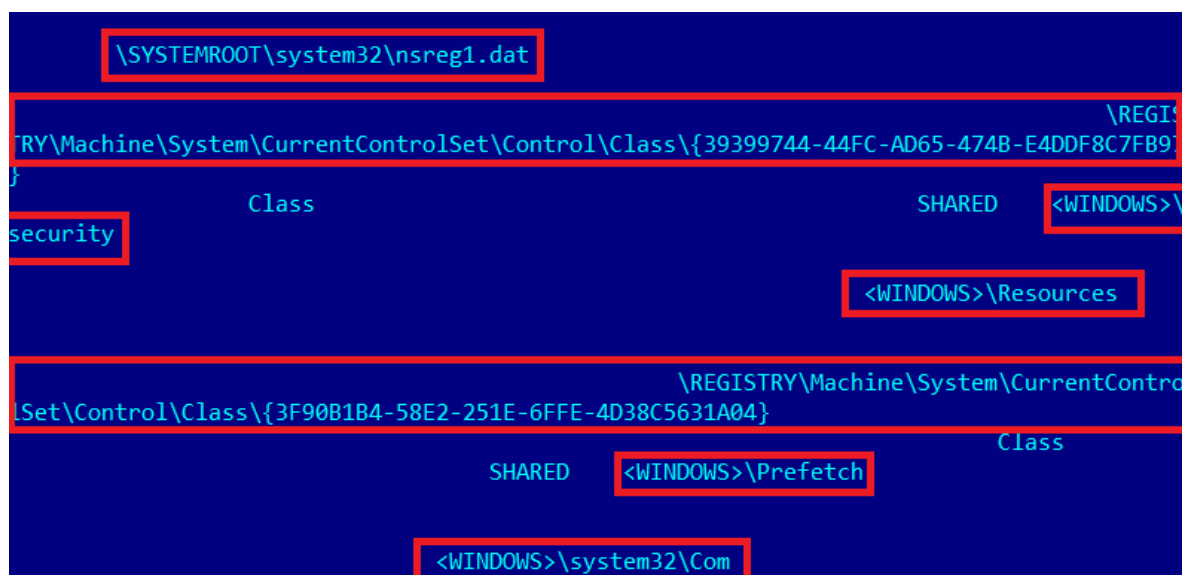
The second stage also creates a marker file that can be used to identify the infected machine. Known filenames for this marker are:

- %SYSTEMROOT%\system32\nsreg1.dat
- %SYSTEMROOT%\system32\bssec3.dat
- %SYSTEMROOT%\system32\msrdc64.dat

These files have their timestamp set to the timestamp of the system file '%SYSTEMROOT%\system32\lsass.exe'

The second stage has additional code for removing the startup code of Regin if signaled by the third stage. Its configuration data contains the locations of the first three stages, including registry keys, names of the directories that hold the encrypted EAs, and the location of the initial driver.

Essentially, the second stage can remove all the Regin stages from the system, effectively cleaning the machine and leaving only the encrypted VFS behind.



*Decrypted configuration block of the second stage*

## Stage 2 – loader – 64-bit

### Known MD5:

d446b1ed24dad48311f287f3c65aeb80

The 64-bit version of the second stage loader is a PE DLL module, since the 64-bit bootstrap chain operates in user mode.

Just like the first stage, it loads the encrypted body of the next stage from the end of the physical disk and decrypts it with a hardcoded RC5 key, then decompresses it using the nr2 algorithm from the UCL library.

After decryption and decompression, the code checks if the next stage is a Windows PE DLL module, and if it is, it loads and executes it.

## Stage 3 – 32-bit – kernel mode manager “VMEM.sys”

### Known MD5s:

8486ec3112e322f9f468bdea3005d7b5

da03648948475b2d0e3e2345d7a9bbbb

On 32-bit systems, the third stage is implemented as a driver module and provides the basic functionality of the malicious framework. It is responsible for operating the encrypted virtual file system and loading additional plugins, and also provides several built-in plugins for the entire framework.

The module initializes the framework, sets up the plugin system and starts the actual work cycle of the malware. It also passes execution to the plugin id **50221** that is loaded from the VFS.

Built-in plugins provided by this module are:

| Id    | Plugin description                                                                |
|-------|-----------------------------------------------------------------------------------|
| 1     | Core framework functionality                                                      |
| 13    | UCL library for compression and decompression using the nrz2 family of algorithms |
| 15    | RC5 encryption and decryption facilities                                          |
| 61    | API for manipulating the encrypted virtual file system (VFS)                      |
| 7     | API for manipulating the encrypted virtual file system (VFS)                      |
| 50225 | API for code injection and kernel-mode hooking                                    |
| 50215 | System information                                                                |
| 50223 | Module notification routines                                                      |
| 50111 | Utilities                                                                         |

## Stage 3 – 64-bit

On 64-bit Windows systems, stage 3 is missing. Stage 2 loads the dispatcher directly from the disk and runs it.

## Stage 4 (32-bit) / 3 (64-bit) – dispatcher module, ‘disp.dll’

32-bit

| Known MD5s:                      |
|----------------------------------|
| 1e4076caa08e41a5befc52efd74819ea |
| 68297fde98e9c0c29cecc0ebf38bde95 |
| 6cf5dc32e1f6959e7354e85101ec219a |
| 885dcd517faf9fac655b8da66315462d |
| a1d727340158ec0af81a845abd3963c1 |

64-bit

| Known MD5:                       |
|----------------------------------|
| de3547375fbf5f4cb4b14d53f413c503 |

The dispatcher library is the user-mode core of the framework. It is loaded directly as the third stage of the 64-bit bootstrap process, or extracted and loaded from the VFS as module 50221 as the fourth stage on 32-bit systems.

It implements a set of internal plugins:

| Id | Plugin description                                                                |
|----|-----------------------------------------------------------------------------------|
| 1  | Core framework functionality                                                      |
| 13 | UCL library for compression and decompression using the nrz2 family of algorithms |
| 15 | RC5 encryption and decryption facilities                                          |

| Id    | Plugin description                                           |
|-------|--------------------------------------------------------------|
| 61    | API for manipulating the encrypted virtual file system (VFS) |
| 7     | API for manipulating the encrypted virtual file system (VFS) |
| 11    | File writer                                                  |
| 51    | Autostart installation routines                              |
| 17    | In-memory storage object                                     |
| 19    | Configuration storage object                                 |
| 50035 | Winsock-based network transport                              |
| 25    | Network transport using packet filters                       |
| 9     | Network transport-related utilities                          |

The dispatcher takes care of the most complicated tasks of the Regin platform, such as providing an API to access virtual file systems, basic communications and storage functions, as well as network transport sub-routines. *In essence, the dispatcher is the brain that runs the entire platform.*

## Stage 4 – Virtual File Systems (32/64-bit)

The most interesting code from the Regin platform is stored in encrypted file storages, known as Virtual File Systems (VFSes).

During our analysis we were able to obtain 24 VFSes from multiple victims around the world. Generally, these have random names and can be located in several places in the infected system:

| Folder on disk                       | File name                                                                              | Description                       |
|--------------------------------------|----------------------------------------------------------------------------------------|-----------------------------------|
| C:\Windows\System32\config\          | SystemAudit.Evt, SystemLog.Evt, SecurityLog.Evt, Security-Audit.Evt, CACHE, SESSIONMGR | Old / ancient style, still around |
| C:\Windows\System32\                 | UsrClass.dat                                                                           | Old / ancient style, still around |
| C:\WINDOWS\pchealth\helpctr\Database | cdata.dat, cdata.edb                                                                   | Old / ancient style, still around |
| C:\Windows\System32\config\          | UsrEvent.evt, ApplicationLog.Evt                                                       | Inside VFS, 6th stage             |
| C:\Windows\Panther\                  | setup.etl.000                                                                          | Used in a 64-bit infection        |
| C:\Windows\System32\wbem\repository\ | INDEX2.DATA, OBJECTS2.DATA                                                             | New style encryption, May 2014    |
| C:\Windows\System32\                 | dnscache.dat, mregnx.dat<br>displn32.dat, dmdskwk.dat, nvwsnu.dat, tapiscfg.dat        | New style encryption, May 2014    |

Each VFS has a structure that is very similar to a real disk file system such as FAT. The VFS files start with a header that provides basic information required to operate the file system. The header is followed by the bitmap of used/free sectors and then by the file table.

| Offset | Size         | Field description                                       |
|--------|--------------|---------------------------------------------------------|
| 00     | 02           | Sector size                                             |
| 02     | 02           | Maximum number of sectors                               |
| 04     | 02           | Maximum number of files                                 |
| 06     | 01           | Unknown                                                 |
| 07     | 04           | CRC32 of first seven bytes of the header with seed 0x45 |
| 0B     | 04           | Size of the file ID field, in bytes                     |
| 0F     | 02           | Number of files                                         |
| 11     | maxSectors/8 | Sector usage bitmap                                     |
|        |              | File table                                              |
|        |              | Sectors                                                 |

Files are described by file table entries:

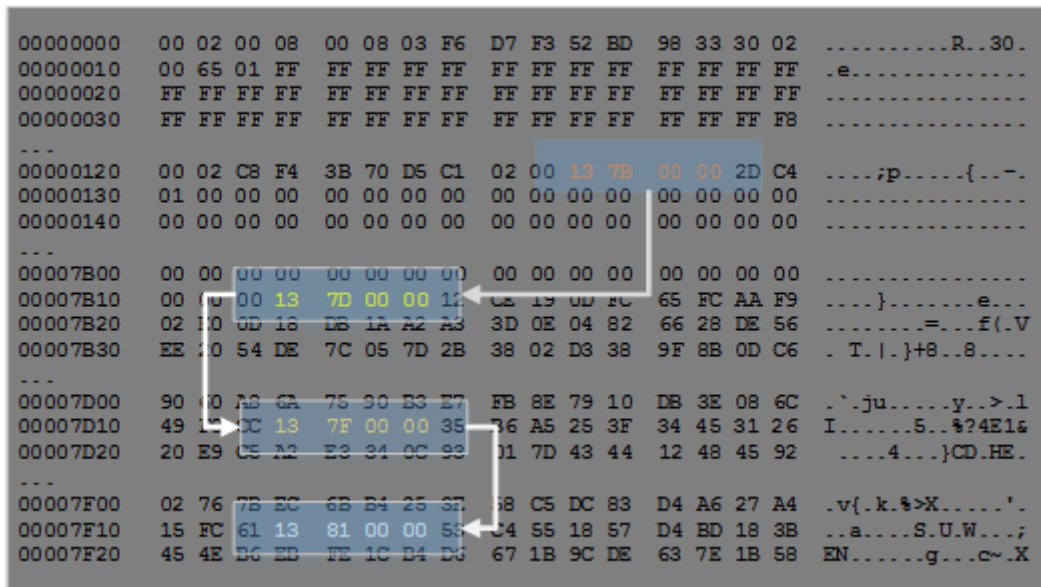
| Offset | Size                      | Field description                     |
|--------|---------------------------|---------------------------------------|
| 00     | 04                        | CRC32 of file contents with seed 0x27 |
| 04     | 04                        | File size                             |
| 08     | 04                        | Offset of the first sector            |
| 0C     | Size of the file ID field | File ID / Plugin ID                   |

Each sector starts with a 32-bit integer that is the offset of the next sector of the file.

|    |                                 |
|----|---------------------------------|
| 00 | Offset of the next sector       |
| 04 | (Sector size)*byte of file data |

An example:

- File record at offset 0x122, file ID 50221, offset of the first sector 0x7B13
- Sector at 0x7B13, next sector at 0x7D13
- Sector at 0x7D13, next sector at 0x7F13,
- Sector at 0x7F13, next sector at 0x8113, etc.

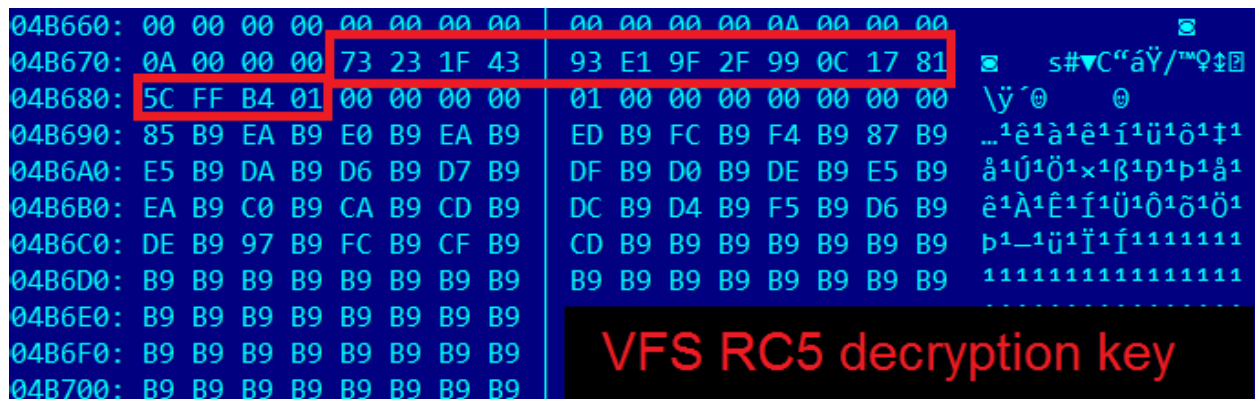


|          |             |             |             |             |                      |
|----------|-------------|-------------|-------------|-------------|----------------------|
| 00000000 | 00 02 00 08 | 00 08 03 F6 | D7 F3 52 BD | 98 33 30 02 | .....R...30..        |
| 00000010 | 00 65 01 FF | FF FF FF FF | FF FF FF FF | FF FF FF FF | .e.....              |
| 00000020 | FF FF FF FF | FF FF FF FF | FF FF FF FF | FF FF FF FF | .....                |
| 00000030 | FF FF FF FF | FF FF FF FF | FF FF FF FF | FF FF FF FF | .....                |
| ---      |             |             |             |             |                      |
| 00000120 | 00 02 C8 F4 | 3B 70 D5 C1 | 02 00 13 7B | 00 00 2D C4 | ....ip.....{...-     |
| 00000130 | 01 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | .....                |
| 00000140 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | .....                |
| ---      |             |             |             |             |                      |
| 00007B00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | .....                |
| 00007B10 | 00 00 00 13 | 7D 00 00 12 | CE 19 0D FC | 65 FC AA F9 | ....}......e...      |
| 00007B20 | 02 00 0D 16 | DB 1A A2 A3 | 3D 0E 04 82 | 66 28 DE 56 | .....=...f(.V        |
| 00007B30 | EE 10 54 DE | 7C 05 7D 2B | 38 02 D3 38 | 9F 8B 0D C6 | . T   . } +8 ..8.... |
| ---      |             |             |             |             |                      |
| 00007D00 | 90 00 A3 CA | 75 30 E3 E7 | FB 8E 79 10 | DB 3E 08 6C | .` .ju.....y...>.l   |
| 00007D10 | 49 12 0C 13 | 7F 00 00 35 | 36 A5 25 3F | 34 45 31 26 | I.....5...%?4E1&     |
| 00007D20 | 20 E9 CE A2 | E3 34 0C 93 | 01 7D 43 44 | 12 48 45 92 | ....4....}CD.HE..    |
| ---      |             |             |             |             |                      |
| 00007F00 | 02 76 7B EC | 6B B4 25 3E | 38 C5 DC 83 | D4 A6 27 A4 | .v{.k.%>X.....'      |
| 00007F10 | 15 FC 61 13 | 81 00 00 55 | C4 55 18 57 | D4 BD 18 3B | ..a....S.U.W....;    |
| 00007F20 | 45 4E DC ED | FE 1C D4 D9 | 67 1B 9C DE | 63 7E 1B 58 | EN.....g....c~.X     |

Example of Regin VFS parsing

Although the structures of the file system are unencrypted, the file entries are encrypted. The encryption algorithm used is RC5, and many records are also compressed using the nrV2e algorithm from the UCL library. UCL is an open source implementation of the proprietary NRV ('Not Really Vanished') compression algorithm, and was originally used by the UPX tool. The reason why the attackers chose UCL is simple: it's small, compact and requires little to no additional memory for decompression.

Each VFS we encountered was encrypted with a 16 bytes key, which can vary from victim to victim. Based on our experience, most files were however encrypted with the same key, **{73 23 1F 43 93 E1 9F 2F 99 0C 17 81 5C FF B4 01}** stored in the dispatcher module or VMEM.sys kernel core.



|         |             |             |             |             |                    |             |             |
|---------|-------------|-------------|-------------|-------------|--------------------|-------------|-------------|
| 04B660: | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00        | 00 00 00 00 | 00 00 00 00 |
| 04B670: | 0A 00 00 00 | 73 23 1F 43 | 93 E1 9F 2F | 99 0C 17 81 | 5C FF B4 01        | 00 00 00 00 | 01 00 00 00 |
| 04B680: | 5C FF B4 01 | 00 00 00 00 | 01 00 00 00 | 00 00 00 00 | 00 00 00 00        | 00 00 00 00 | 00 00 00 00 |
| 04B690: | 85 B9 EA B9 | E0 B9 EA B9 | ED B9 FC B9 | F4 B9 87 B9 | ...1ê1à1ê1í1ü1ô1†1 |             |             |
| 04B6A0: | E5 B9 DA B9 | D6 B9 D7 B9 | DF B9 D0 B9 | DE B9 E5 B9 | â1Û1Ö1×1ß1Ð1þ1â1   |             |             |
| 04B6B0: | EA B9 C0 B9 | CA B9 CD B9 | DC B9 D4 B9 | F5 B9 D6 B9 | ê1À1Ê1Î1Û1Ô1ö1Ï1   |             |             |
| 04B6C0: | DE B9 97 B9 | FC B9 CF B9 | CD B9 B9 B9 | B9 B9 B9 B9 | þ1—1ü1ÿ1ÿ11111111  |             |             |
| 04B6D0: | B9 B9 B9 B9 | B9 B9 B9 B9 | B9 B9 B9 B9 | B9 B9 B9 B9 | 1111111111111111   |             |             |
| 04B6E0: | B9 B9 B9 B9 | B9 B9 B9 B9 | B9 B9 B9 B9 | B9 B9 B9 B9 |                    |             |             |
| 04B6F0: | B9 B9 B9 B9 | B9 B9 B9 B9 | B9 B9 B9 B9 | B9 B9 B9 B9 |                    |             |             |
| 04B700: | B9 B9 B9 B9 | B9 B9 B9 B9 | B9 B9 B9 B9 | B9 B9 B9 B9 |                    |             |             |

**VFS RC5 decryption key**

VFS RC5 decryption key inside the dispatcher module (disp.dll)

In all, we observed about a dozen different VFS keys.

The following plugins were observed inside the VFSes we collected. These are all identified by a 16-bit number. The plugins are referenced by these numbers; they are like filenames on a normal file system and allow the dispatcher to easily load or reference them.

The binary modules are referenced by these numbers as plugin identifiers and usually have similar internal DLL names; e.g., the plugin with ID '50121' will have the internal name '50121.dll' in its export table. Compressed binary modules are accompanied by binary files with the same ID. These files contain the size of the decompressed module and are not included in the description.

Known data blocks and their configuration IDs:

|       |                                                                                                          |
|-------|----------------------------------------------------------------------------------------------------------|
| 0     | 4 bytes, unknown                                                                                         |
| 1     | Configuration data; timestamped binary data                                                              |
| 4     | 4 bytes, unknown                                                                                         |
| 9     | Transport list and configurations, including peer hostnames and addresses                                |
| 11    | Location of an additional '.evt' file, usually 'ApplicationLog.evt'                                      |
| 13    | Configuration data                                                                                       |
| 14    | Configuration data                                                                                       |
| 15    | Peer encryption keys                                                                                     |
| 19    | Peer network configuration data                                                                          |
| 25    | Packet filter configuration                                                                              |
| 51    | 4 bytes, unknown                                                                                         |
| 10001 | Strings: 'legspinv2.6', 'WILLISCHECKv2.0', additional configuration data                                 |
| 10207 | Configuration data                                                                                       |
| 10404 | Configuration data                                                                                       |
| 10405 | Configuration data                                                                                       |
| 10505 | 1 byte; unknown                                                                                          |
| 50009 | Configuration data                                                                                       |
| 50013 | List of processes ('snort.exe', 'wireshark.exe', 'rundll32.exe', etc.)                                   |
| 50049 | Log of GSM base station commands. Very rare, most interesting                                            |
| 50079 | Location of a temporary file                                                                             |
| 50121 | Drive names                                                                                              |
| 50139 | Event log provider names                                                                                 |
| 50181 | Data used by network transport plugins                                                                   |
| 50185 | Plugin configuration                                                                                     |
| 50227 | Plugin configuration                                                                                     |
| 50233 | Process file name list (Explorer.exe, VMWareService.exe, Update.exe, Msiexec.exe, MailService.exe, etc.) |
| 56001 | Plugin configuration                                                                                     |
| 57003 | Configuration data                                                                                       |

Known executable modules and their plugin IDs:

|   |                                                |
|---|------------------------------------------------|
| 0 | Data only                                      |
| 1 | Core framework functionality                   |
| 3 | '3.sys' file timestamp manipulation; utilities |

|       |                                                                                              |
|-------|----------------------------------------------------------------------------------------------|
| 9     | Network transport-related utilities                                                          |
| 15    | RC5 encryption and decryption facilities                                                     |
| 25    | Network transport using packet filters                                                       |
| 27    | ICMP network listener using raw sockets                                                      |
| 10001 | Command-line data collection and administration tools                                        |
| 10105 | Utilities                                                                                    |
| 10107 | User logon and impersonation, user and domain name collection                                |
| 10207 | 'pp.dll' Keylogger and clipboard sniffer                                                     |
| 10211 | Network share enumeration and manipulation                                                   |
| 10309 | Pipe/mailslot backend for plugin 10207                                                       |
| 10405 | Timestamp conversions                                                                        |
| 10507 | Extraction from the protected storage and credential storages                                |
| 11101 | Detection of process hooks, directory enumeration                                            |
| 11701 | Collects information about connected USB storage devices, creates storage files              |
| 20005 | Driver installation/removal routines                                                         |
| 20027 | Collects information about sessions, installed browsers and proxy settings                   |
| 20029 | Remote registry manipulation routines                                                        |
| 20073 | Interception of system network drivers                                                       |
| 50001 | File system data collection and manipulation                                                 |
| 50011 | File data extraction                                                                         |
| 50013 | Searches for potentially dangerous processes by module path/name (sniffers, debuggers, etc.) |
| 50015 | Retrieves current system time in Unix timestamp format                                       |
| 50017 | Time-related utilities                                                                       |
| 50019 | Sniffer using a packet filter                                                                |
| 50025 | System information, network share enumeration and scans                                      |
| 50029 | Sniffer utilities                                                                            |
| 50033 | Event log hooks                                                                              |
| 50035 | Winsock-based network transport                                                              |
| 50037 | Network transport over HTTP                                                                  |
| 50047 | Sniffer utilities                                                                            |
| 50049 | HTTP/SMTP/SMB credentials sniffer                                                            |
| 50051 | Network transport over HTTPS                                                                 |
| 50053 | Sniffer utilities                                                                            |

|       |                                                                                                              |
|-------|--------------------------------------------------------------------------------------------------------------|
| 50061 | Utilities                                                                                                    |
| 50063 | BPF filter parser                                                                                            |
| 50073 | Network routing utilities                                                                                    |
| 50079 | Temporary file manipulation                                                                                  |
| 50081 | Network transport and configuration utilities                                                                |
| 50097 | DNS sniffer                                                                                                  |
| 50101 | Extended system information; task scheduler data                                                             |
| 50113 | Utilities                                                                                                    |
| 50115 | NDIS filter                                                                                                  |
| 50117 | Network information: connections, adapters, DNS cache, statistics                                            |
| 50121 | File system traversal                                                                                        |
| 50123 | HTTPS server, 'Microsoft-IIS/6.0'                                                                            |
| 50139 | Windows event log reader                                                                                     |
| 50185 | Dumping users' password hashes (LM database)                                                                 |
| 50211 | Driver hooking and hook detection                                                                            |
| 50215 | 'BEEP' driver, used by the 50211 plugin                                                                      |
| 50219 | Injects plugins in processes                                                                                 |
| 50221 | 'disp.dll' user-mode core of the framework                                                                   |
| 50223 | Module notification routines                                                                                 |
| 50225 | API for code injection and kernel-mode hooking                                                               |
| 50227 | Code injection and hooking utilities                                                                         |
| 50231 | Replication using network shares and local persistence, remote filename used: 'ADMIN\$\SYSTEM32\SVCSTAT.EXE' |
| 50233 | Plugin injection utilities                                                                                   |
| 50251 | Keyboard driver hooking                                                                                      |
| 50271 | Network transport over SMB (named pipes)                                                                     |
| 55001 | E-mail message extraction module 'U_STARBUCKS'                                                               |
| 55011 | MS Exchange data extraction, appointment information                                                         |
| 55007 | POP3 proxy server, used in conjunction with plugin 55001                                                     |
| 56001 | Winsock networking routines                                                                                  |

The attackers can dynamically add and delete plugins inside the VFS and each victim installation has a different set of plugins depending on the type of activity the attackers need to execute. For example, only some of the VFSes we have seen had lateral movement modules, designed for infecting other computers in the network.

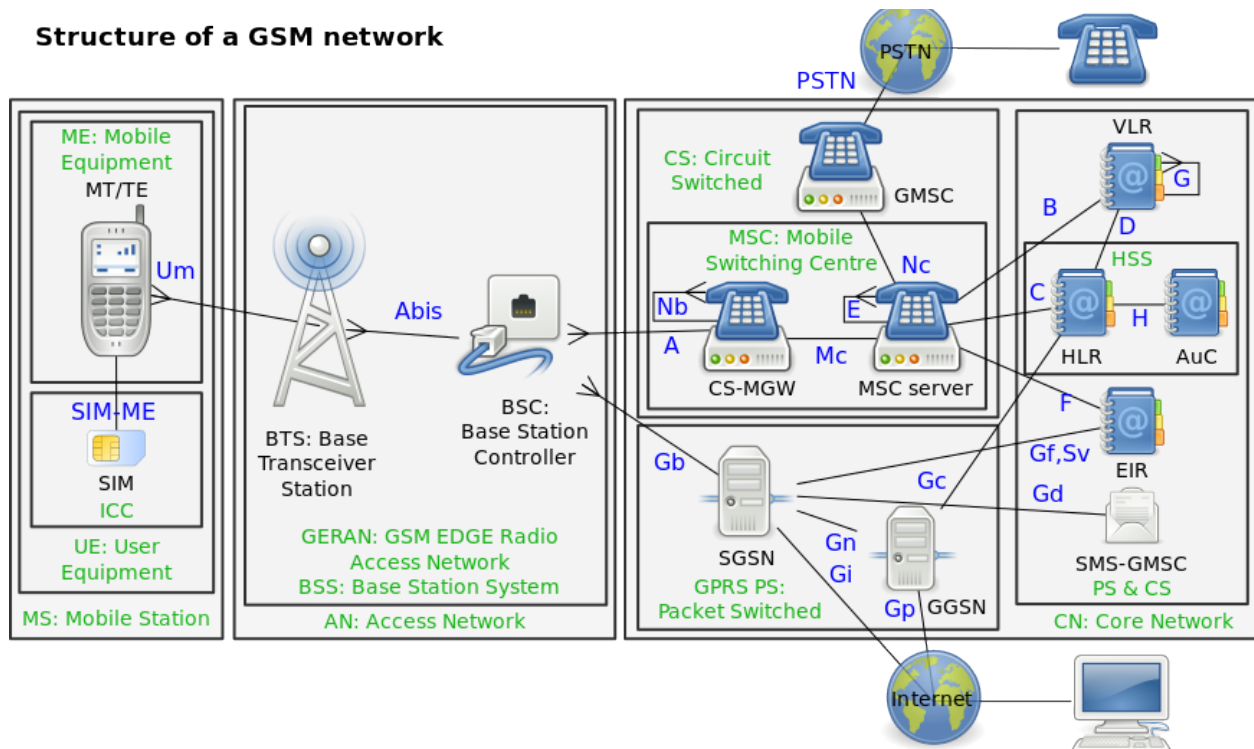


```
\0 ▶ T0 ▶ L0 ▶ @0 ▶ <0 ▶ 40 ▶ 00 ▶ (0 ▶ L0 ▶ ▶0 ▶ CURRENT=
VERSION= TMARK= LM= __utma= TM= PREF=ID= UID= GRID= TK=UID= USERID= php
Ads_id= PHPSESSID= ASP.NET_SessionId= LASTVISIT= TIMESET= TW=WINKER= SMSWAP=
SESSID 0 ; %s%s;s;%s% Cookie: yyyÿŸ◀ ▶f◀ ▶ yyyÿÿi ▶ō ▶ yyyÿ/H ▶
3H ▶shit yyyÿKa ▶fa ▶G l o b a l \ e t h e r n e t L o w e r R a n g e
I n t e r f a c e s N d i E x p o r t S Y S T E M \ C u r r e n t C o n t r o l
l s e t \ C o n t r o l \ C l a s s \ { 4 D 3 6 E 9 7 2 - E 3 2 5 - 1 1 C E - B F C 1 -
0 8 0 0 2 B E 1 0 3 1 8 } M P F _ B i n d S y s t e m \ C u r r e
n t C o n t r o l S e t \ S e r v i c e s \ T c p i p \ L i n k a g e \ \ . \ G l o b
a l \ % s \ \ . \ % s yyyÿ4l ▶8l ▶ c l o s e % Proxy-Connection: Connecti
on: Content-Length: Basic Digest NTLM Proxy-Authenticate: CONNECT
GET WORKGROUP CONNECT %s:%d HTTP/1.0 %sHost: %s %s %s %s %s %s Content-Length: 0 % GET
Service Pack Group Start ErrorControl Type
System \ CurrentControlSet \ Services \ NtLoadDr
iver NtUnloadDriver ControlSet \ Enum \ Root \ LEGACY _
\ Services \ SeLoadDriverPrivilege ntddl \ Reg
istry \ Machine \ System \ CurrentControlSet \ Servi
ces \ yyyÿÿv▶iv▶ yyyÿÿEX▶IX▶ yyyÿÿi▶]▶<]▶%x:%x:%x:%x:%x:%x:%x:%x:c
%d.%d.%d.%d% shit DWX SSM yyyÿöü▶ü▶ yyyÿ[-▶v-▶ yyyÿöæ▶iæ▶255.255.255.
255 Global \ ethernet LowerRange Interfaces
Ndi Export SYSTEM \ CurrentControlSet \ Control
\ Class \ { 4 D 3 6 E 9 7 2 - E 3 2 5 - 1 1 C E - B F C 1 - 0 8 0 0 2 B E 1 0 3 1 8
} M P F _ B i n d S y s t e m \ C u r r e n t C o n t r o l S e t
```

## GSM targeting

The most interesting aspect we have found so far regarding RegIn relates to an infection of a large GSM operator. One VFS encrypted entry we located had internal id 50049.2, and appears to be an activity log on a GSM Base Station Controller.

### Structure of a GSM network



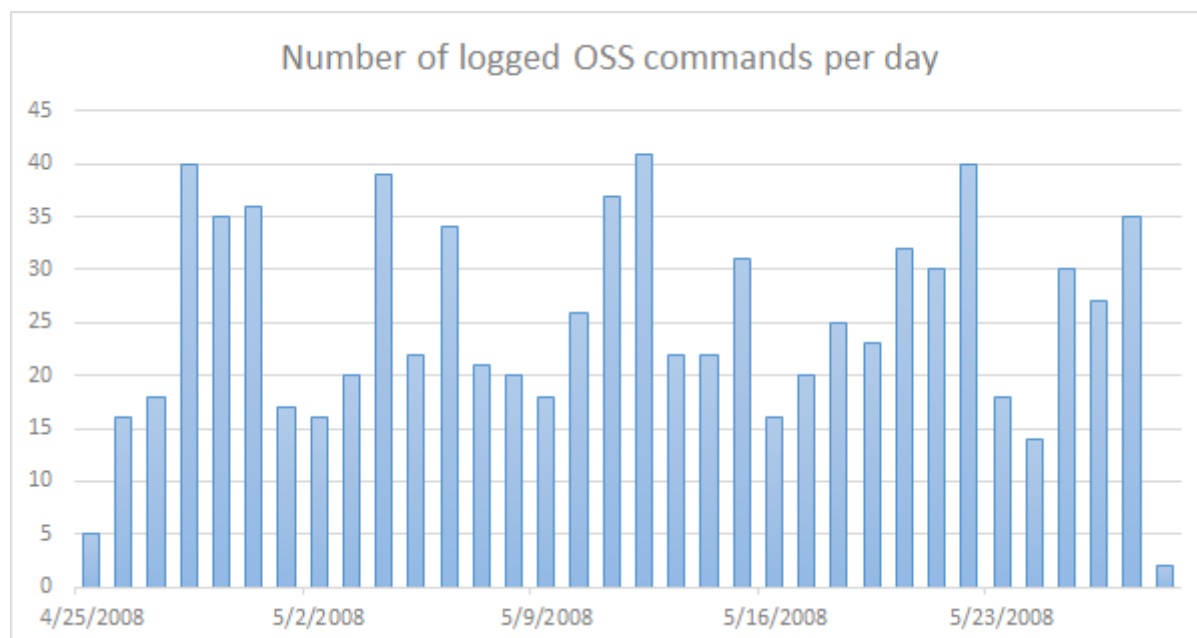
From [https://en.wikipedia.org/wiki/Base\\_station\\_subsystem](https://en.wikipedia.org/wiki/Base_station_subsystem)

According to the GSM documentation (<http://www.telecomabc.com/b/bsc.html>): 'The Base Station Controller (BSC) is in control of and supervises a number of Base Transceiver Stations (BTS). The BSC is responsible for the allocation of radio resources to a mobile call and for the handovers that are made between base stations under his control. Other handovers are under the control of the MSC.'

Here's a look at the decoded RegIn GSM activity log:

```
00: 01 00 00 04 00 00 03 01 | 00 8A 51 49 FA B1 A6 C8 00 00 00 00 00 00 00 00
10: 01 30 00 00 00 00 00 00 | 00 00 00 6F 73 73 00 00 00 00 00 00 00 00 00 00
20: 0D 0A 4E 65 77 00 00 00 | 0D 0A 0D 0A 32 00 00 00 00 00 00 00 00 00 00 00
30: 0D 0A 6D 6D 6C 0D 0A 72 | 6C 63 72 70 3A 63 65 6C 00 00 00 00 00 00 00 00
40: 6C 3D 61 6C 6C 3B 0D 00 | 03 01 00 7E 30 10 37 C5 00 00 00 00 00 00 00 00
50: A6 C8 01 46 00 00 00 00 | 00 00 00 68 00 00 00 00 00 00 00 00 00 00 00 00
60: 65 64 00 0A 0D 0A 0D 0A | 42 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
70: 40 0D 0A 0D 0A 0D 0A 6D | 6D 6C 0D 0A 72 78 6D 6F 00 00 00 00 00 00 00 00
80: 70 3A 6D 6F 74 79 3D 72 | 78 6F 74 72 78 3B 0D 00 00 00 00 00 00 00 00
90: 03 01 00 66 D4 A8 A5 CB | A6 C8 01 46 00 00 00 00 00 00 00 00 00 00 00 00
A0: 00 00 00 00 00 00 00 00 | 65 64 61 00 0A 0D 0A 0D 0A 0D 0A 0D 0A 0D 0A 0D 0A
```

This log is about 70KB in size and contains hundreds of entries like the ones above. It also includes timestamps that indicate exactly when the command was executed.



The entries in the log appear to contain Ericsson OSS MML (Man-Machine Language as defined by ITU-T) commands (see [https://en.wikipedia.org/wiki/Operations\\_support\\_system](https://en.wikipedia.org/wiki/Operations_support_system)).

Here's a list of some commands issued on the Base Station Controller, together with some of their timestamps:

```
2008-04-25 11:12:14: rxmop:moty=rxotrx;
2008-04-25 11:58:16: rxmsp:moty=rxotrx;
2008-04-25 14:37:05: rlcpr:cell=all;
2008-04-26 04:48:54: rxble:mo=rxocf-170,subord;
2008-04-26 06:16:22: rxtcp:MOty=RXOtg,cell=kst022a;
2008-04-26 10:06:03: IOSTP;
2008-04-27 03:31:57: rlstc:cell=pty013c,state=active;
2008-04-27 06:07:43: allip:acl=a2;
2008-04-28 06:27:55: dtstp:DIP=264rb12;
2008-05-02 01:46:02: rlstp:cell=all,state=halted;
2008-05-08 06:12:48: rlmfc:cell=NGR035W,mbcchno=83&512&93&90&514&522,listtype=active;
2008-05-08 07:33:12: rlnri:cell=NGR058y,cellr=ngr058x;
2008-05-12 17:28:29: rrtpp:trapool=all.
```

Descriptions for the commands:

**rxmop** - check software version type;

**rxmsp** - list current call forwarding settings of the Mobile Station;

**rlcpr** - list off call forwarding settings for the Base Station Controller;

**rxble** - enable (unblock) call forwarding;

**rxtcp** - show the Transceiver Group of particular cell;

**allip** - show external alarm;

**dtstp** - show Digital Path (DIP) settings (DIP is the name of the function used for supervision of the connected PCM (Pulse Code Modulation) lines);

**rlstc** - activate cell(s) in the GSM network;

**rlstp** - stop cell(s) in the GSM network;  
**rlmfc** - add frequencies to the active broadcast control channel allocation list;  
**rlnri** - add cell neighbor;  
**rrtpp** - show radio transmission transcoder pool details.

The log seems to contain not only the executed commands but also usernames and passwords of some engineering accounts:

```
sed[snip]:Alla[snip]  
hed[snip]:Bag[snip]  
oss:New[snip]  
administrator:Adm[snip]
```

In total, the log indicates that commands were executed on 136 different cells. Some of the cell names include '**prn021a, gzn010a, wdk004, kbl027a, etc...**'. The command log we obtained covers a period of about one month, from April 25, 2008 through May 27, 2008. It is unknown why the commands stopped in May 2008 though; perhaps the infection was removed or the attackers achieved their objective and moved on. Another explanation is that the attackers improved or changed the malware to stop saving logs locally and that is why only some older logs were discovered.

## Communication and C&C

The C&C mechanism implemented in Regin is extremely sophisticated and relies on communication drones deployed by the attackers throughout the victim networks. Most victims communicate with another machine in their own internal network through various protocols as specified in the config file. These include HTTP and Windows network pipes. The purpose of such a complex infrastructure is to achieve two goals: (i) to give attackers access deep into the network, potentially bypassing air gaps; and (ii) to restrict as much as possible the traffic to the C&C.

Here's a look at the decoded configurations::

```
17.3.40.101 transport 50037 0 0 y.y.y.5:80 ; transport 50051 217.y.y.yt:443  
17.3.40.93 transport 50035 217.x.x.x:443 ; transport 50035 217.x.x.x:443  
50.103.14.80 transport 27 203.199.89.80 ; transport 50035 194.z.z.z:8080  
51.9.1.3 transport 50035 192.168.3.3:445 ; transport 50035 192.168.3.3:9322  
18.159.0.1 transport 50271 DC ; transport 50271 DC
```

In the above table we see configurations extracted from several victims that bridge together infected machines in what appears to be virtual networks: 17.3.40.x, 50.103.14.x, 51.9.1.x, 18.159.0.x. One of these routes reaches out to the 'external' C&C server at **203.199.89.80**.

The numbers right after the 'transport' indicate the plugin that handles the communication. These are in our case:

- 27 - ICMP network listener using raw sockets
- 50035 - Winsock-based network transport
- 50037 - Network transport over HTTP
- 50051 - Network transport over HTTPS
- 50271 - Network transport over SMB (named pipes)

The machines located on the border of the network act as routers, effectively connecting victims from inside the network with C&Cs on the Internet.

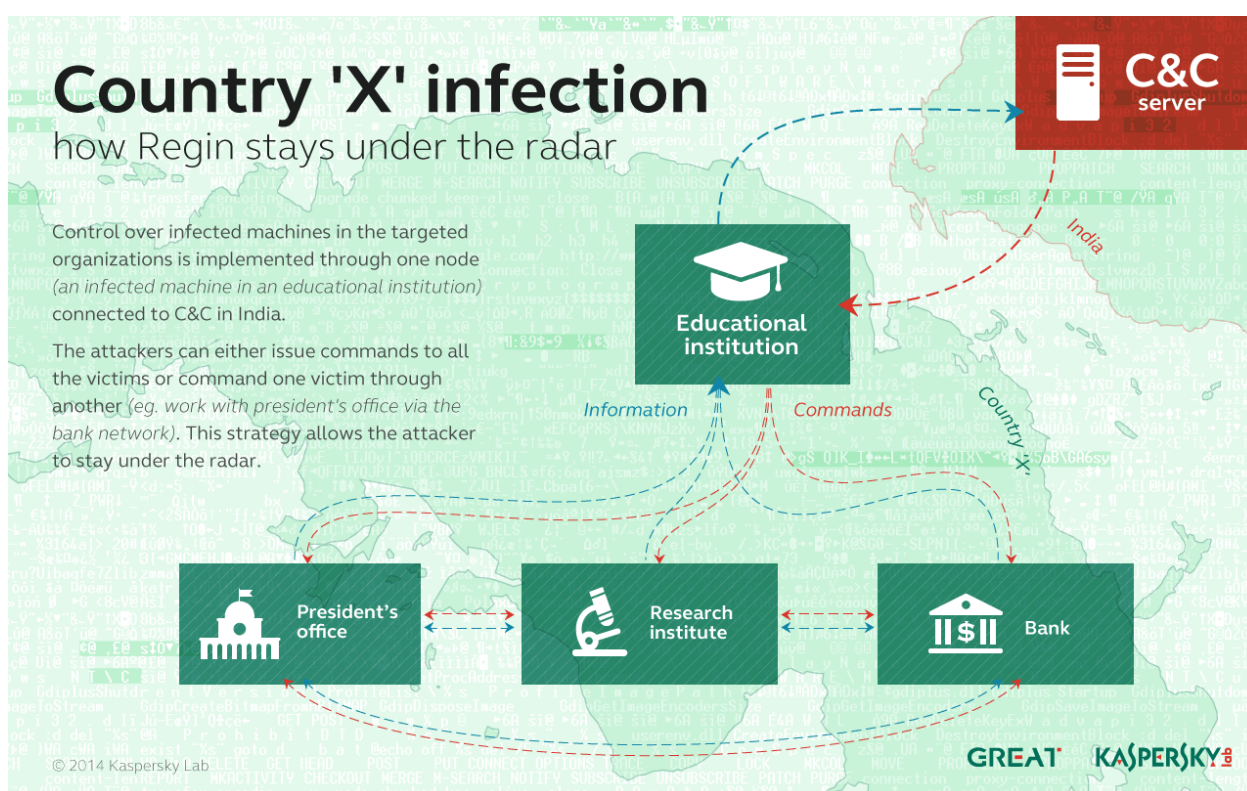
After decoding all the configurations we have collected, we were able to identify the following external C&Cs.

| C&C server IP   | Location          | Description                            |
|-----------------|-------------------|----------------------------------------|
| 61.67.114.73    | Taichung, Taiwan  | Chwbn                                  |
| 202.71.144.113  | Chetput, India    | Chennai Network Operations (team-m.co) |
| 203.199.89.80   | Thane, India      | Internet Service Provider              |
| 194.183.237.145 | Brussels, Belgium | Perceval S.a.                          |

One particular case includes a country in the Middle East. It was rather astonishing, so we thought it should be mentioned. In this country all the victims we identified communicate with each other, forming a peer-to-peer network. The P2P network includes **the president's office, a research center, an educational institution network and a bank**.

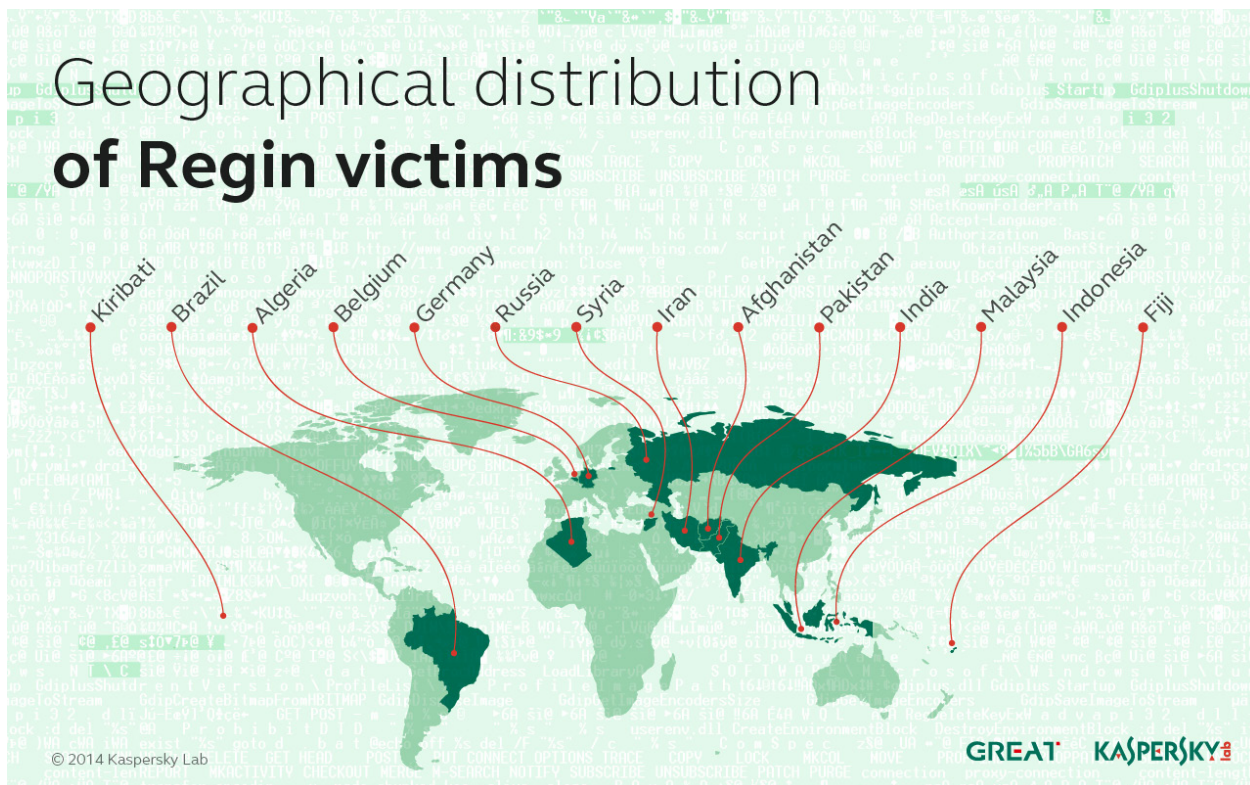
Spread across the country, these victims are all interconnected with each other. One of the victims contains a translation drone, which has the ability to forward packets outside the country, to the C&C in India.

This represents a rather interesting command-and-control mechanism, which is guaranteed to raise little suspicion. For instance, if all commands to the president's office are sent through the bank's network, then all the malicious traffic visible to the president's office sysadmins will only be with the bank, in the same country.



## Victim statistics

Over the past two years we have been collecting statistics on the attacks and victims of Regin. These were aided by the fact that even after the malware is uninstalled, certain artifacts are left behind, which can help identify an infected (but cleaned) system. For instance, we have seen several cases where the systems were cleaned but the 'msrdc64.dat' infection marker was left behind.



So far, victims of Regin have been identified in **14 countries**:

- Afghanistan
- Algeria
- Belgium
- Brazil
- Fiji
- Germany
- India
- Indonesia
- Iran
- Kiribati
- Malaysia
- Pakistan
- Russia
- Syria

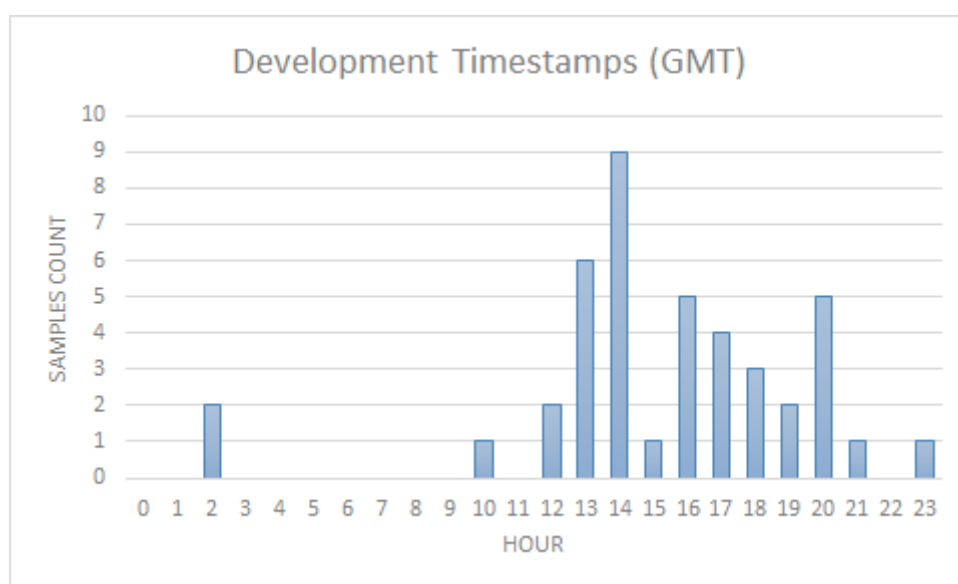
In total, we counted **27 different victims**, although it should be pointed out that the definition of a victim here refers to a full entity, including its entire network. The number of unique PCs infected with Regin is of course much, much higher.

From the map above, **Fiji** and **Kiribati** are unusual, because we rarely see such advanced malware in such remote, tiny countries. In particular, the victim in Kiribati is most unusual. To put this into context, Kiribati is a small island in the Pacific with a population around 100,000. According to experts, Kiribati is probably going to become one of the first victims of global warming, as it will be under water by 2050. (<http://www.businessinsider.com/pacific-island-nation-kiribati-sinking-2014-5?op=1>)

## Attribution

Considering the complexity and cost of Regin's development, it is likely that this operation is supported by a nation state. While attribution remains a very difficult problem when it comes to professional attackers such as the ones behind Regin, certain metadata extracted from the samples is still worth a look.

We have collected timestamps from samples, which are normally put automatically by the development software:



As this information could be easily altered by the developers, it is up to the reader to attempt to interpret this: as an intentional false flag, or a non-critical indicator left by the developers.

More information about Regin is available to Kaspersky Intelligent Services' clients. Contact: [intelreports@kaspersky.com](mailto:intelreports@kaspersky.com)

## Conclusions

For more than a decade, a sophisticated group known as Regin has targeted high-profile entities around the world with an advanced malware platform. As far as we can tell, the operation is still active, although the malware may have been upgraded to more sophisticated versions. The most recent sample we have seen was from a 64-bit infection. This infection was still active in the spring of 2014.

The name Regin is apparently a switched around 'In Reg', short for 'In Registry', as the malware can store its modules in the registry. This name and the detections first appeared in anti-malware products around March 2011.

In some ways the platform reminds us of another sophisticated malware: Turla (<http://securelist.com/analysis/publications/65545/the-epic-turla-operation/>). Some similarities include the use of virtual file systems and the deployment of communication drones to bridge networks together. Yet through their implementation, coding methods, plugins, hiding techniques and flexibility, *Regin surpasses Turla as one of the most sophisticated attack platforms* we have ever analyzed.

The ability of this group to penetrate and monitor GSM networks is perhaps the most unusual and interesting aspect of these operations. In today's world, we have become too dependent on cellphone networks that rely on ancient communication protocols with little or no security available for the end user. Although all GSM networks have mechanisms embedded that allow entities such as law enforcement to track suspects, there are other parties which can gain this ability and then abuse it to launch other types of attacks against mobile users.

Kaspersky Lab products detect modules from the Regin platform as: Trojan.Win32.Regina.gen and Rootkit.Win32.Regina.

If you detect a Regin infection in your network, contact us at: [intelservices@kaspersky.com](mailto:intelservices@kaspersky.com)

## Technical appendix and indicators of compromise:

### Yara rules:

```
rule apt_regin_vfs {
meta:
  copyright = "Kaspersky Lab"
  description = "Rule to detect Regin VFSes"
  version = "1.0"
  last_modified = "2014-11-18"
strings:
  $a1={00 02 00 08 00 08 03 F6 D7 F3 52}
  $a2={00 10 F0 FF F0 FF 11 C7 7F E8 52}
  $a3={00 04 00 10 00 10 03 C2 D3 1C 93}
  $a4={00 04 00 10 C8 00 04 C8 93 06 D8}
condition:
  ($a1 at 0) or ($a2 at 0) or ($a3 at 0) or ($a4 at 0)
}
rule apt_regin_dispatcher_disp_dll {
meta:
  copyright = "Kaspersky Lab"
  description = "Rule to detect Regin disp.dll dispatcher"
  version = "1.0"
  last_modified = "2014-11-18"
strings:
  $mz="MZ"
  $string1="shit"
  $string2="disp.dll"
  $string3="255.255.255.255"
  $string4="StackWalk64"
  $string5="imagehlp.dll"
condition:
  ($mz at 0) and (all of ($string*))
}
rule apt_regin_2013_64bit_stagel {
meta:
  copyright = "Kaspersky Lab"
  description = "Rule to detect Regin 64 bit stage 1 loaders"
  version = "1.0"
  last_modified = "2014-11-18"
  filename="wshnetc.dll"
  md5="bddf5afbea2d0eed77f2ad4e9a4f044d"
  filename="wsharp.dll"
  md5="c053a0a3f1edcbbfc9b51bc640e808ce"
strings:
  $mz="MZ"
  $a1="PRIVHEAD"
  $a2="\\\\.\\PhysicalDrive%d"
  $a3="ZwDeviceIoControlFile"
condition:
  ($mz at 0) and (all of ($a*)) and filesize < 100000
}
```

```
rule apt_regin_2011_32bit_stagel {
meta:
  copyright = "Kaspersky Lab"
  description = "Rule to detect Regin 32 bit stage 1 loaders"
  version = "1.0"
  last_modified = "2014-11-18"
strings:
  $key1={331015EA261D38A7}
  $key2={9145A98BA37617DE}
  $key3={EF745F23AA67243D}
  $mz="MZ"
condition:
  ($mz at 0) and any of ($key*) and filesize < 300000
}
rule apt_regin_rc5key {
meta:
  copyright = "Kaspersky Lab"
  description = "Rule to detect Regin RC5 decryption keys"
  version = "1.0"
  last_modified = "2014-11-18"
strings:
  $key1={73 23 1F 43 93 E1 9F 2F 99 0C 17 81 5C FF B4 01}
  $key2={10 19 53 2A 11 ED A3 74 3F C3 72 3F 9D 94 3D 78}
condition:
  any of ($key*)
}
```

## MD5s:

### Stage 1 files, 32 bit:

```
06665b96e293b23acc80451abb413e50
187044596bc1328efa0ed636d8aa4a5c
1c024e599ac055312a4ab75b3950040a
2c8b9d2885543d7ade3cae98225e263b
4b6b86c7fec1c574706cecedf44abded
6662c390b2bbbd291ec7987388fc75d7
b269894f434657db2b15949641a67532
b29ca4f22ae7b7b25f79c1d4a421139d
b505d65721bb2453d5039a389113b566
26297dc3cd0b688de3b846983c5385e5
ba7bb65634ce1e30c1e5415be3d1db1d
bfbe8c3ee78750c3a520480700e440f8
d240f06e98c8d3e647cbf4d442d79475
ffb0b9b5b610191051a7bdf0806e1e47
```

### Unusual stage 1 files apparently compiled from various public source codes merged with malicious code:

```
01c2f321b6bfdb9473c079b0797567ba
47d0e8f9d7a6429920329207a32ecc2e
744c07e886497f7b68f6f7fe57b7ab54
db405ad775ac887a337b02ea8b07fddc
```

**Stage 1, 64-bit system infection:**

```
bddf5afbea2d0eed77f2ad4e9a4f044d
c053a0a3f1edcbbfc9b51bc640e808ce
e63422e458afdfe111bd0b87c1e9772c
```

**Stage 2, 32 bit:**

```
18d4898d82fcb290dfed2a9f70d66833
b9e4f9d32ce59e7c4daf6b237c330e25
```

**Stage 2, 64 bit:**

```
d446b1ed24dad48311f287f3c65aeb80
```

**Stage 3, 32 bit:**

```
8486ec3112e322f9f468bdea3005d7b5
da03648948475b2d0e3e2345d7a9bbbb
```

**Stage 4 32 bit:**

```
1e4076caa08e41a5befc52efd74819ea
68297fde98e9c0c29cecc0ebf38bde95
6cf5dc32e1f6959e7354e85101ec219a
885dcd517faf9fac655b8da66315462d
a1d727340158ec0af81a845abd3963c1
```

**Stage 4 64 bit:**

```
de3547375fbf5f4cb4b14d53f413c503
```

Note: Stages 2,3 and 4 do not appear on infected systems as real files on disk. Hashes are provided for research purposes only.

**Registry branches used to store malware stages 2 and 3:**

- \REGISTRY\Machine\System\CurrentControlSet\Control\RestoreList
- \REGISTRY\Machine\System\CurrentControlSet\Control\Class\{39399744-44FC-AD65-474B-E4DDF-8C7FB97}
- \REGISTRY\Machine\System\CurrentControlSet\Control\Class\{3F90B1B4-58E2-251E-6FFE-4D38C5631A04}
- \REGISTRY\Machine\System\CurrentControlSet\Control\Class\{4F20E605-9452-4787-B793-D0204917CA58}
- \REGISTRY\Machine\System\CurrentControlSet\Control\Class\{9B9A8ADB-8864-4BC4-8AD5-B17DFDBB9F58}

**C&C IPs:**

|                 |                                                      |
|-----------------|------------------------------------------------------|
| 61.67.114.73    | Taiwan, Province Of China Taichung Chwbn             |
| 202.71.144.113  | India Chetput Chennai Network Operations (team-m.co) |
| 203.199.89.80   | India Thane Internet Service Provider                |
| 194.183.237.145 | Belgium Brussels Perceval S.a.                       |

## VFS RC5 decryption algorithm

This algorithm is used throughout the code and is referenced as RC5 in the document, although the implementation and the way the cipher is invoked is specific to Reginfo.

The implementation in C++ follows:

```
void RC5Decrypt(uint8_t* rc5Key, uint8_t* data, size_t len)
{
    uint8_t      iv[8];
    rc5_ctx_t     ctx;
    uint8_t*     encrypted;
    size_t       encryptedLen;

    rc5_init(rc5Key, 128, 20, &ctx);
    memcpy(iv, data, 8);
    encrypted = data + 8;
    encryptedLen = len - 8;
    if ( encryptedLen % 8 )
    {
        uint8_t    ivLocal[8];
        if ( encryptedLen < 8 )
            memcpy(ivLocal, iv, 8);
        else
            memcpy(ivLocal, encrypted + encryptedLen - (encryptedLen % 8) - 8, 8);
        rc5_enc(ivLocal, &ctx);
        for (size_t idx = 0; idx < (encryptedLen % 8); idx++)
            encrypted[idx + encryptedLen - (encryptedLen % 8)] ^= ivLocal[idx];
    }
    if ( encryptedLen / 8 > 1 )
    {
        for (ssize_t blockIdx = (encryptedLen / 8) - 1; blockIdx > 0; blockIdx--)
        {
            rc5_dec(encrypted + blockIdx*8, &ctx);
            for (size_t idx = 0; idx < 8; idx++)
                encrypted[blockIdx*8 + idx] ^= encrypted[(blockIdx-1)*8 + idx];
        }
    }
    if ( encryptedLen / 8 > 0 )
    {
        rc5_dec(encrypted, &ctx);
        for (size_t idx = 0; idx < 8; idx++)
            encrypted[idx] ^= iv[idx];
    }
}
```

## Kaspersky Lab HQ

39A/3 Leningradskoe Shosse  
Moscow, 125212  
Russian Federation

[more contact details](#)

Tel: +7-495-797-8700

Fax: +7-495-797-8709

E-mail: [info@kaspersky.com](mailto:info@kaspersky.com)

Website: [www.kaspersky.com](http://www.kaspersky.com)

The Kaspersky Lab logo, featuring the word "KASPERSKY" in a bold, sans-serif font, followed by a stylized shield icon containing a cross.